VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce dopravních značek z obrazů kamer zabudovaných ve vozidlech (kruhové značky)

Traffic Signs Detection from the Cameras Built in Vehicles (Circular Signs)

2010 Jiří Vjater

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2010 .....

Rád bych na tomto místě poděkoval všem, kteří přispěli svou radou ke vzniku této práce

Abstrakt

Tato práce se zabývá především detekcí a rozpoznáváním kruhových dopravních značek. V první části jsou krátce shrnuty jednotlivé možnosti. Druhá část je pak věnována implementaci systému rozpoznávajícího dopravní značky.

Klíčová slova: detekce dopravních značek, rozpoznání dopravních značek, Houghovy transformace, momentové invarianty, diplomová práce

Abstract

This thesis focuses mainly on detection and recognition of circular road signs. The first section briefly summarizes the various options. The second part is devoted to implementation of the system rozpoznávajícího traffic signs.

Keywords: traffic signs detection, traffic signs recognition, Hough transform, moment invariants, diploma thesis

Seznam použitých zkratek a symbolů

21HT	–	21 Hough Transform
LVQ	–	Learning Vector Quantization

Obsah

1 Úvod

S rozvojem elektronických systémů pro podporu řízení vozidel, se začaly objevovat i snahy o rozpoznávání dopravních značek. Protože dopravní značky představují přesně definovanou množinu obrazových vzorů, jedná se na první pohled o velmi jednoduchý úkol. Existuje však několik faktorů, které celý problém výrazně ztěžují. Několik ukázek je na obrázku 1.

Prvním z těchto problémů je orientace dopravní značky. V případu kruhových dopravních značek nejsou k dispozici body, podle kterých by šlo před vlastním rozpoznáním značku otočit. Na první pohled se sice tento problém jeví jako nepodstatný, protože každá značka má své předepsané otočení, v praxi však vlivem náklonu silnice nebo špatného uchycení může dojít k otočení.

Dále je třeba se vypořádat s různými podmínkami osvětlení. Dnes nejčastěji používaný RGB barevný model nemá separovanou jasovou složku. Jasová informace je tedy zakódována ve všech barevných kanálech dohromady. Z tohoto důvodu nelze s tímto barevným modelem bez jakýchkoliv dalších úprav pracovat.

Vlivem povětrnostních podmínek může také docházet ke korozi materiálu nebo vyblednutí jednotlivých barev. Tím dochází i ke změně kontrastu a snížené čitelnosti dopravních značek.

V neposlední řadě mají na dopravní značky vliv i vandalové. V jejich případě, stejně jako u koroze a vyblednutí barev, bohužel nelze provést během procesu rozpoznávání žádné korekce.

Některé z výše uvedených problémů je možné eliminovat nebo jejich vliv minimalizovat. Tato práce obsahuje dvě základní části. V první se nachází shrnutí již implementovaných systémů pro rozpoznávání dopravních značek. Druhá část je věnována vlastní implementaci, která částečně vychází z těchto dřívějších prací, a jejímu testování.

Obrázek 1: Ukázky vlivů osvětlení a poškozených značek

2 Dopravní značky

Dopravní značky obdobné těm dnešním začaly vznikat na počátku dvacátého století. Z této doby pocházejí i první návrhy na standardizaci dopravního značení. Ty první byly mezinárodního charakteru. Pozdější úmluvy měly většinou platnost pouze v rámci Evropy. Proto jsou si evropské dopravní značky podobné. Tato podobnost se týká především tvaru, barevného lemování a většiny symbolů.

Dopravní značky můžeme podle jejich významu rozdělit na výstražné, zákazové, příkazové, informativní a upravující přednost v jízdě. Každá z těchto kategorií má své typické barevné provedení. Například pro výstražné a zákazové značky je typické červené lemování.

Dále lze dopravní značky rozdělit podle jejich tvaru na trojúhelníkové, čtyřúhelníkové, kruhové a jiné. Jednotlivé tvary opět převážně spadají do několika málo významových tříd. Například kruhový tvar mají všechny zákazové a příkazové značky, do tříd upravujících přednost v jízdě a informativních značek patří pouze po jedné kruhové značce.

Dopravní značka obsahuje několik základních identifikačních prvků:

1. Symbol
2. Tvar značky
3. Lemování
4. Pole

Symbol a tvar značky udávají její význam. Symbolem může být piktogram, čísla nebo text. Význam může udávat dokonce i absence symbolu (například značka pro zákaz vjezdu).

Lemování se vyskytuje pouze u zákazových a výstražných značek a značek upravujících přednost v jízdě. Pomáhá rozlišit tyto kategorie od značek příkazových a informativních. Zpravidla mívá červenou barvu. Lemování může dopravní značku rozdělit na několik polí.

Pole je jednobarevný prostor uvnitř lemování mimo symbol. Pokud lemování na značce neexistuje, jedná se o jednobarevný prostor mimo vlastní symbol. Podle této definice má pole stejný informativní význam jako symbol a lemování dohromady. Pole bývá vždy barevně výrazně odlišeno od symbolu a lze ho tedy rozeznat i na snímcích ve stupních šedi. Tedy dokonce i rozpoznávat jednotlivé značky.

3 Přehled dříve navrhovaných implementací

Dříve navrhované systémy pro rozpoznávání dopravních značek lze v zásadě rozdělit na systémy, které využívají barevnou informaci k detekci nebo rozpoznávání značek, a systémy, které se bez této informace obejdou. Metody pracující s barevnými obrazy většinou požadují předzpracování vstupního obrazu. Tyto úpravy barev se pokusím shrnout v kapitole 3.1. Účelem tohoto předzpracování je snížit počet bodů, ve kterých se může hledaná značka nacházet.

Detekční část může být pro obě skupiny různá, může totiž být závislá na předchozím zpracování.

Systém vyhodnocení je většinou pro obě skupiny shodný. Výjimkou jsou pouze systémy, které využívají barev i ke zjištění, o kterou značku se jedná.

3.1 Předzpracování barevných obrazů

Hlavním problémem těchto systémů je závislost RGB modelu na osvětlení. Tato závislost není lineární. Nelze tedy jednoznačně prahovat jednotlivé složky, aby došlo k roztřídění barev podle požadavků. Opět existuje několik přístupů, které tuto závislost odbourávají.

Nejjednodušší možností je převod do barevného modelu, ve kterém je jasová složka oddělena. Mezi nejpoužívanější modely patří HSI nebo HSL. Nevýhodou modelů HSI a HSL je příliš široké množství možných hodnot pro vyjádření bílé barvy. Ta je potřebná pro značky Konec nejvyšší povolené rychlosti, Konec zákazu předjíždění, Konec zákazu předjíždění pro nákladní automobily, Konec zákazu zvukových výstražných znamení a Konec všech zákazů. Všechny tyto značky obsahují pouze bílé pole a v něm černý nebo šedý symbol.

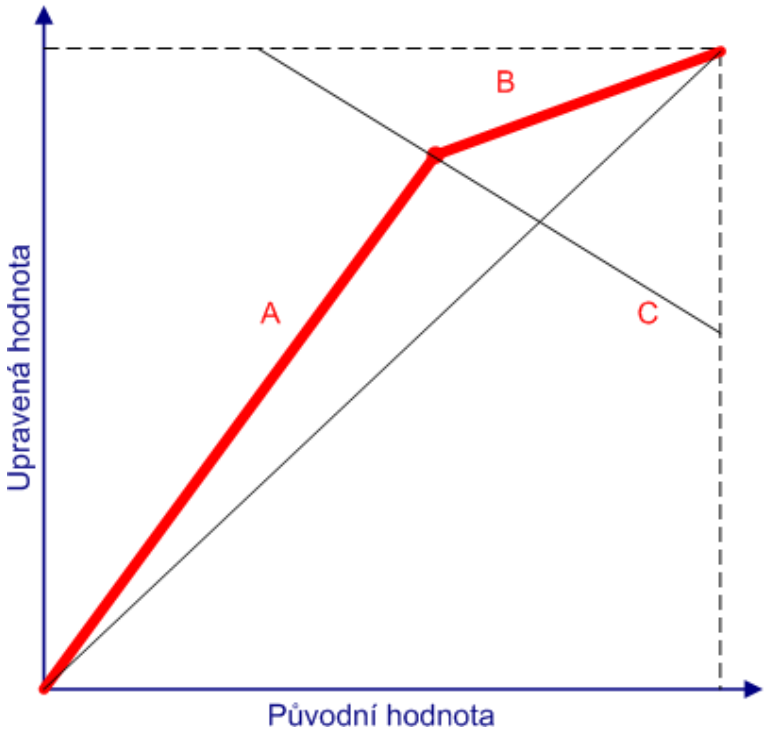
Během návrhu systému [8] byl zkoušen i barevný model YUV. Zároveň však bylo vyzkoušeno prahování jednotlivých barevných složek RGB modelu v závislosti na ostatních složkách. Toto řešení se ukázalo jako rychlejší a jednodušší. Vztahy pro prahování jsou následující:

$$g(x,y) = 1 \begin{cases} \alpha_{min} \cdot f_g(x,y) \leq f_r(x,y) \leq \alpha_{max} \cdot f_g(x,y) \\ \beta_{min} \cdot f_b(x,y) \leq f_r(x,y) \leq \beta_{max} \cdot f_b(x,y) \\ \gamma_{min} \cdot f_b(x,y) \leq f_g(x,y) \leq \gamma_{max} \cdot f_b(x,y) \end{cases} \tag{1}$$

Zároveň byla navržena úprava tónové křivky k vyrovnání vlivů barevného světla, které vzniká například při západu a východu slunce. Ideální by sice byla úprava podle bílého referenčního bodu, ale takovýto bod se v reálných snímcích špatně vyhledává. Z tohoto důvodu se vychází z předpokladu, že cesta je šedé barvy. Na cestě se provede obdélníkový výběr a provede se součet v každém barevném kanálu zvlášť. Kanál jehož součet, není ani minimální ani maximální, se nebude upravovat. Ostatní se upraví podle rovnic 2, 3 a 4, které popisují obrázek 2.

$$A : y = \alpha \cdot x \tag{2}$$

$$B : y = \beta \cdot x + \gamma \tag{3}$$



Obrázek 2: Úprava tónové křivky

$$C : y = -x + k \tag{4}$$

Pro upravované kanály se parametr α vypočte podle rovnice 5. Funkce $f_{ref}(x, y)$ je jasovou funkcí kanálu, který byl zvolen jako neměnný. Funkce $f_{upravovany}(x, y)$ je jasovou funkcí upravovaného kanálu. Parametr k, na kterém je závislý parametr β, se volí experimentálně. Parametr γ se spočte jako průsečík A a C. Hodnota β se zvolí tak, aby úsečka B končila v bodě $(255, 255)$. Uvedené operace je nutné provést pro oba upravované kanály.

$$\alpha = \frac{\sum_x \sum_y f_{ref}(x, y)}{\sum_x \sum_y f_{upravovany}(x, y)} \tag{5}$$

V článku [7] se navrhuje prahovat poměry hodnot barevných složek vůči referenční barevné složce. Ve svém návrhu autoři zvolili jako referenční červený kanál. Prahování se tedy provádí podle funkce 6. V případech, které nejsou v dané rovnici uvedeny, funkce $g(x, y)$ nabývá hodnotu k_2.

$$g(x, y) = k_1 \begin{cases} R_a \leq f_r(x, y) \leq R_b \\ TG_a \leq \frac{f_g(x,y)}{f_r(x,y)} \leq TG_b \\ TB_a \leq \frac{f_b(x,y)}{f_r(x,y)} \leq TB_b \end{cases} \tag{6}$$

Hodnoty R_a, R_b, TG_a, TG_b, TB_a a TB_b jsou prahové hodnoty pro jednotlivé barevné složky. Funkce $f_r(x, y)$,$f_r(x, y)$ a $f_b(x, y)$ jsou hodnoty jednotlivých barevných kanálů na

Barva	R	G	B
Černá	0	0	0
Modrá	0	0	255
Zelená	0	255	0
Azurová	0	255	255
Červená	255	0	0
Purpurová	255	0	255
Žlutá	255	255	0
Bílá	255	255	255

Tabulka 1: Přehled normalizovaných barev

souřadnicích (x, y). Konstanty k_1 a k_2 jsou hodnotami, kterých má nabývat prahovaný obraz.

Dalším z možných přístupů k problému různého osvětlení je normalizace barev. Tato metoda byla popsána v článku [9]. Každá z barev v rozích RGB prostoru představuje jednu z barev po normalizaci. V obraze tedy zůstanou barvy uvedené v tabulce 1. K převodu se opět využije jednoduché prahování pro jednotlivé barevné složky. Pokud je prahovaná složka vyšší než práh, nastaví se její hodnota na 255. Tím dojde k převodu na barvy uvedené v tabulce 1. Prahy se volí podle následujícího postupu.

Vstupní obrazy se podle této metody dělí na příliš světlé, normální a příliš tmavé. Za příliš světlé se považují obrazy, jejichž střední hodnota jasu jednotlivých pixelů je větší než 180. Naopak příliš tmavé obrazy mají střední hodnotu jasu pixelů menší než 100. Pro příliš světlé obrazy se doporučuje volit prahy $(180, 130, 130)$, pro příliš tmavé $(40, 30, 30)$ a pro ostatní $(70, 75, 60)$.

Z hlediska českých kruhových dopravních značek jsou z takto upravených barev významné bílá, modrá, červená, purpurová a azurová. Poslední dvě barvy jsou do výběru zahrnuty, protože u těchto barev nebylo přesně rozhodnuto, jestli je červená nebo modrá složka dominantní.

Nejzajímavějším přístupem je použití vícevrstvé neuronové sítě v článku [10], k určení jestli má daný pixel správnou barvu pro další zpracování.

Výhodou uvedených řešení je snížení počtu obrazových bodů pro další zpracování. Navíc je možno využít metody, které by jinak nepřipadaly v úvahu. Nevýhodou může být vznik vyššího počtu hran vlivem odebrání některých barev nebo převodem na jiné barvy. Při zpracování takovýchto obrazů pomocí Houghových transformací výrazně narůstá časová složitost.

3.2 Detekce značek

Pokud byl obraz dříve prahován, je možné ho rozdělit na jednotlivé objekty. A u nich pak zjišťovat, jestli odpovídají požadovanému tvaru. Toto vyhodnocení může být prováděno neuronovou sítí jako například v článku [10]. Další možností je využití vertikálních

a horizontálních histogramů, jak bylo uvedeno v práci [11]. Tvary jako kruh, čtverec nebo trojúhelník mají tyto charakteristiky dostatečně unikátní. Obecně je možno použít jakoukoliv techniku pro rozpoznávání obrazu.

Pro systémy, ve kterých neprobíhá předzpracování, se velmi často používají Houghovy transformace. Nevýhodou tohoto algoritmu je vyšší časová složitost. Naopak zase podává poměrně dobré výsledky.

Další možností je využít detekci rohů pomocí konvoluce, jejíž jádro a jeho výpočet je popsán v článku [7]. Tu je možné použít stejně jako Houghovy transformace ve všech případech. Pomocí detekce rohů je možno nalézt jak trojúhelníky a čtverce, tak dokonce i kruhy.

3.3 Rozpoznávání značek

Velmi častou metodou je použití vícevrstvé neuronové sítě, jejímž vstupem je celá značka. Každý vstupní obraz musí být nejdříve převeden na předem stanovené rozměry odpovídající počtu vstupů neuronové sítě. Pokud je zvolen příliš velký rozměr může neuronová síť obsahovat hodně neuronů, což vede ke snížení výkonnosti. Na druhou stranu se příliš malé obrazy špatně rozpoznávají. Takovéto klasifikátory se pomalu učí jednotlivé vzory, ale rychle vyhodnocují vstupy. Protože je vyhodnocovací proces závislý na rotaci rozpoznávaného objektu, je nutné učit vzory v několika rotacích.

Často se také používá korelace. Pro trojúhelníkové značky se provádí v barycentrických souřadnicích, pro čtvercové značky v kartézské souřadné soustavě a pro kruhové značky v polárních souřadnicích. Obrazy je opět nutno převést na předem stanovenou velikost. Z hlediska složitosti je takovéto vyhodnocování vhodné u trojúhelníkových a čtvercových značek. Důvodem je malý počet možností otočení objektu. U kruhových dopravních značek není žádný referenční bod kromě středu a ten nevypovídá nic o její rotaci. Proto je nutno prozkoumat všechny přípustné rotace, čili v polární soustavě souřadnic posuny po ose úhlu. Takovýto postup však může být poměrně náročný na výpočetní výkon.

Poslední možností, kterou bych zde rád uvedl, je využití momentových invariantů. Ty nejsou závislé na rotaci objektu a není tedy nutné prověřovat veškerá možná natočení značky. Problémy mohou nastat v případě, že jeden symbol v různém natočení odpovídá odlišné dopravní značce. V takovýchto případech je třeba dopočítat úhel natočení vůči hlavní ose objektu. Momentové invarianty je třeba dále vyhodnotit. K tomuto účelu lze použít pravděpodobnostní neuronovou síť, vícevrstvou neuronovou síť nebo pouze vzdálenost od předpočtených hodnot. Každá z uvedených metod má své výhody i nevýhody.

Všechny uvedené algoritmy pro rozpoznávání dopravních značek mohou pracovat s binárním obrazem. S barevnými obrazy mohou pracovat pouze neuronové sítě a korelace. Momentové invarianty mohou být použity také pro obrazy ve stupních šedi.

4 Realizovaná metoda

Program pracuje především s barevnými snímky. Je tomu tak kvůli binarizaci, která je jedním z kroků během zpracování snímku. Při použití obrazů ve stupních šedi, se může stát, že její výsledek nebude odpovídající.

Dále jsem se snažil zvolit implementaci, která by byla nezávislá na natočení dopravní značky podle vodorovné osy ve směru kolmém k rozpoznávané značce. Tím došlo k vyloučení neuronových sítí k přímému rozpoznávání obrazu. Dalším požadavkem je rychlost porovnávání všech možných natočení se vzory. Jako vhodné se jeví použití momentových invariantů, které jsou nezávislé na natočení obrazu. Oproti korelaci u nich není nutné provádět posuny nebo rotace.

Jak je znázorněno na obrázku 3, navrhovaný systém lze podle funkcí rozdělit do několika základních částí, kterými při zpracování musí projít každý vstupní snímek.

Prvním blokem je detekce kruhů. Jejím úkolem je vybrat ze snímku oblasti, ve kterých se nachází kruhy a tedy i značky. Výstupem jsou oblasti, ve kterých se může nacházet značka. Celý tento blok je nezávislý na barevné informaci a pracuje pouze s jasovou složkou.

V části sloužící k binarizaci se tyto oblasti převádí do takového formátu, aby byly jasně zřetelné charakteristické symboly v každé značce. Zároveň tato část slouží k minimalizaci vlivů osvětlení. Výsledek operací prováděných v této části významně ovlivňuje přesnost dalších výpočtů. Vzhledem k problémům s binarizací obrazu pouze na základě jasu, se v této části programu používají i barevné snímky.

Z binarizovaného obrazu se vypočtou momentové charakteristiky, které jsou invariantní vůči rotaci. Ty se následně porovnají s předpočtenými hodnotami a vybere se odpovídající vzor.

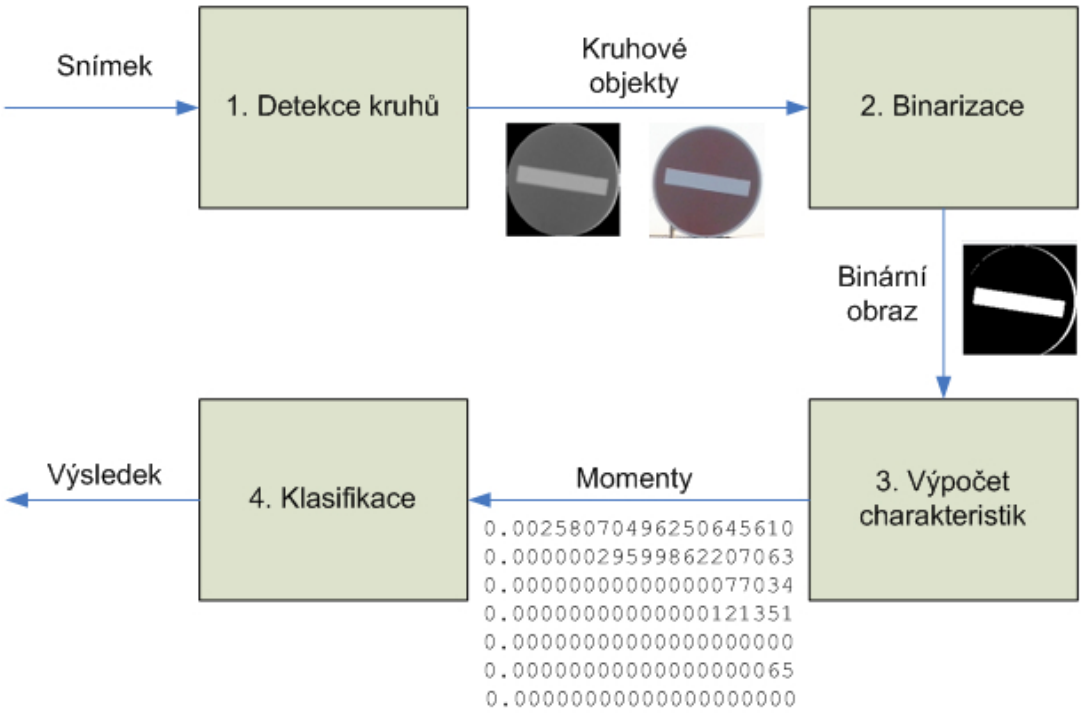
Následně je třeba na základě vypočtených charakteristik určit, o který z rozpoznávaných vzorů se jedná.

4.1 Detekce kruhů

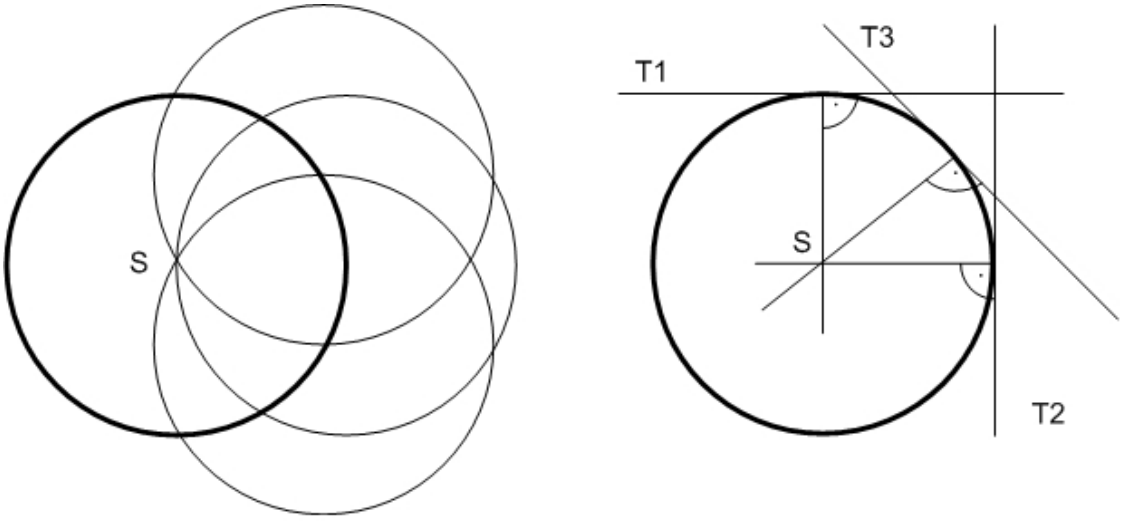
Prvním úkolem je zjištění oblastí, ve kterých se nachází kruhové objekty a tedy možná i kruhové značky. Pro tento účel jsem zvolil Houghovy transformace. Výhodou tohoto algoritmu je odolnost vůči přerušení hranice kruhu a v některých případech i schopnost detekovat mírně eliptické objekty. Naopak nevýhodou je vyšší výpočetní náročnost.

4.1.1 Houghovy transformace

Klasický algoritmus Houghových transformací pro hledání kruhů je založen na tom, že kružnice se stejným poloměrem a se středem na stejné kružnici se všechny protnou v jediném bodě, ve středu kružnice na níž jsou středy jednotlivých kružnic. Geometrické provedení tohoto vyhledávání středu kružnice je znázorněno na obrázku 4 vlevo. Jak je vidět na uvedeném obrázku tento postup negeneruje průsečíky jen ve středu kružnice, jejíž střed je vyhledáván, ale i mimo ni.



Obrázek 3: Blokové schéma



Obrázek 4: Geometrický význam standartního algoritmu Houghových transformací (vlevo) a algoritmu 21HT (vpravo)

Přístup používaný v počítačové grafice je obdobný tomuto geometrickému modelu. Kružnici nahrazují body hran získaných z původního obrazu a hledají se možné středy. Princip spočívá ve „vykreslování“ kružnic do trojrozměrného akumulátoru. Dva rozměry odpovídají souřadnicím x a y, třetí poloměru kruhů. Pro každý bod ležící na hraně o souřadnicích (x, y) a poloměr r se inkrementují v akumulátoru hodnoty na souřadnicích daných kružnicí (x, y, r). Po provedení této operace pro všechny možné středy a poloměry se v každé rovině x,y vyhledají lokální maxima. Ty se prahují podle zvolené hodnoty, čímž se sníží počet falešných detekcí. Zbylá maxima odpovídají středům kruhů, přičemž poloměrem je souřadnice r určující příslušnou rovinu akumulátoru.

V implementaci OpenCv je použita varianta označovaná jako 21HT. Geometrický význam algoritmu je znázorněn na obrázku 4 vpravo. Nevykreslují se kružnice, ale pouze kolmice k tečnám kružnice jejíž střed hledáme. Jak je z obrázku patrné, průsečíky jsou generovány pouze v hledaném středu.

V této verzi algoritmu je akumulátor pouze dvojrozměrný. Nacházejí se v něm osy x,y. Z původního obrazu se získají nejen hrany ale i jejich gradienty $grad\,(x, y)$, ty určují směr ke středu potenciálního kruhu. Pro každý bod (x, y) ležící na hraně se inkrementují v akumulátoru hodnoty na úsečce dané počátkem (x, y), směrem $grad\,(x, y)$ a délkou $maxr$ odpovídající maximálnímu povolenému poloměru hledaného kruhu. Poté se opět vyhledávají lokální maxima vyšší než zvolený práh. Ty stejně jako v klasickém algoritmu odpovídají středům kruhů. Protože v akumulutáru není žádný záznam o poloměrech, je třeba ho dopočíst. Pro každý ze středů se spočtou vzdálenosti k jednotlivým hranám a vybere se ta nejčastější.

Výhodou této varianty je nižší paměťová náročnost vzhledem k tomu že akumulátor je pouze dvojrozměrný. Po nalezení středů ho lze dokonce smazat. Zároveň výpočet souřadnic na úsečce je rychlejší než výpočet bodů kružnice. Tato metoda je naopak citlivější na prahovou hodnotu akumulátoru. Při příliš nízkém prahu dochází k růstu výpočetní náročnosti v části výpočtu poloměru. Další význačnou vlastností je to, že pokud jsou ve zpracovávaném obraze soustředné kruhy, je vždy vybrán ten větší.

4.1.2 Použitá implementace

Algoritmy uvedené v předchozí kapitole pracují s vysokou přesností, pokud je vzdálenost mezi objekty větší než maximální hledaný poloměr. V opačných případech, především u verze 21HT, dochází k vysokému počtu falešných detekcí. Nezbývá tedy nic jiného než nastavit prahovací hodnotu akumulátoru dostatečně vysoko podle počtu hran v obraze. Problémem jsou však vysoké rozdíly v počtech hran nejen mezi jednotlivými snímky, ale dokonce i mezi různými oblastmi v nich.

Z tohoto důvodu se provádí jako předzpracování rozmazání pro odstranění šumu. To je třeba provádět s ohledem na kvalitu zpracovávaného snímku. Kvalita fotografií, především zaostření a rozložení jasu, bývá běžně podstatně lepší než v případě videa. Na fotografiích tedy vzniká větší počet hran než v případě videozáznamu. Dále je v případě fotografií možný mnohem větší rozdíl mezi minimálním a maximálním poloměrem. S oběmi druhy je tedy třeba zacházet různě. Pro fotografie je postačující Gaussovské roz-

mazání. V případě snímku z videozáznamu, který není sám o sobě příliš zaostřen, však takovéto rozmazání potlačí příliš mnoho hran. Vhodnější je tedy použít mediánový filtr.

Obě tyto úpravy však nejsou úplně postačující. Proto je nutné najít metodu, která sníží počty falešných detekcí.

Jednou ze zkoušených možností je provádět Houghovy transformace hierarchicky. V první fázi se detekují oblasti, ve kterých se kruh může nacházet. V této části může být použito i nižší rozlišení akumulátoru pro zvýšení rychlosti, neboť upřesnění se provádí až v následujícím kroku. V druhé části se v nalezených oblastech potvrzuje přítomnost kruhu, přičemž se předpokládá, že poloměr je téměř shodný s nalezeným. Práh musí být zvolen podstatně nižší než v první fázi, protože je zde méně hran než v první fázi. Tato metoda sama o sobě však neposkytuje dokonalé výsledky především z hlediska časové složitosti, neboť kandidátů z první fáze může být velké množství.

Druhou vyzkoušenou možností je rozdělit obraz do několika oblastí a vyhledávat v každé zvlášť. To umožní definovat práh pro každou část samostatně v závislosti na počtu hran. Rozměry oblasti by měly být větší nebo rovny maximálnímu možnému průměru hledaného kruhu. Sousední oblasti by se měly navzájem překrývat alespoň o největší možný poloměr kruhu. Takto bude vždy více než polovina kruhu v jedné z oblastí. Může se ovšem stát, že některý kruh bude detekován vícekrát, přičemž středy ani poloměry nemusí být nutně totožné.

Prahovací funkce by měla být závislá na počtu hran ve zpracovávané oblasti. Tento počet počet by měl být vztažen k celkové ploše prohledávané oblasti. Dále by bylo vhodné, aby byla přímo úměrná minimálnímu poloměru. Tím se dosáhne toho, aby pro kruhy s malými poloměry a tedy i s malými obvody byl práh dostatečně nízký. Pro velké minimální poloměry pak bude práh naopak vysoký.

Počet průsečíků generovaných Houghovými transformaci však neroste lineárně s počtem bodů hran. Tento problém nastává především u fotografií.

Vyjděme tedy z náledující úvahy. Každé bodu hrany je přiřazena jedna přímka. K vytvoření jednoho průsečíku jsou zapotřebí dvě přímky. Jestliže je v obraze h přímek, je tedy možné za předpokladu, že se přímky nepřekrývají, vytvořit celkem $C_2(h)$ průsečíků.

$$C_2(h) = \frac{h!}{2(h-2)!} \tag{7}$$

$$C_2(h) = \frac{h(h-1)(h-2)!}{2(h-2)!} \tag{8}$$

$$C_2(h) = \frac{h(h-1)}{2} \tag{9}$$

V ideálním případě se tyto průsečíky navrší v jediném bodě nebo se rozdělí na celý prostor. Může se však stát, že orientace přímek daná gradienty v bodech hran, kterými prochází, bude taková, že vytvoří mnoho průsečíků i v místech, která neodpovídají středům hledaných kruhů.

Proto je zapotřebí aby i prahovací funkce rostla s přibývajícím počtem hran rychleji. Není však možné nastavit rovnou počet průsečíků, ty totiž mohou být rozprostřeny po

celé ploše. Prahovací funkce tedy musí být mnohem menší než funkce 9. Zároveň však musí být větší než alespoň dvojnásobek minimálního poloměru.

Počet průsečíků roste rychleji hlavně s vyššími řády. Zároveň roste i šance, že se průsečíky budou hromadit. Po sérii pokusů na testovacích fotografiích jsem zvolil exponenciální funkci, jejíž mocninu tvoří logaritmus o základu deset.

Pro jednodušší manipulaci s celou funkcí je vhodné přidat pevný práh a upravit váhu závislosti na počtu bodů hran pomocí konstanty.

Z těchto úvah jsem došel ke tvaru 10, kde d je pevně stanovená prahová hodnota v závislosti na rozlišení, $P(h)$ pravděpodobnost že náhodně vybraný bod oblasti leží na hraně, $minr$ je minimální hledaný poloměr, a je konstanta ovlivňující množství závislosti prahovací funkce na počtu hran, h je počet bodů náležejících hranám v oblasti a funkce $c(h)$ převádí počet hran na pomaleji rostoucí metriku.

$$y = d + a \cdot b^{c(h)} \cdot minr^2 \cdot 4 \cdot P(h) \tag{10}$$

Tato funkce se však ukázala jako naprosto nevyhovující pro snímky pořízené z videozáznamu, především kvůli jejich kvalitě. Pro takové snímky je vhodná následující funkce:

$$y = 2 \cdot minr + minr \cdot \frac{maxr - minr}{maxr} \tag{11}$$

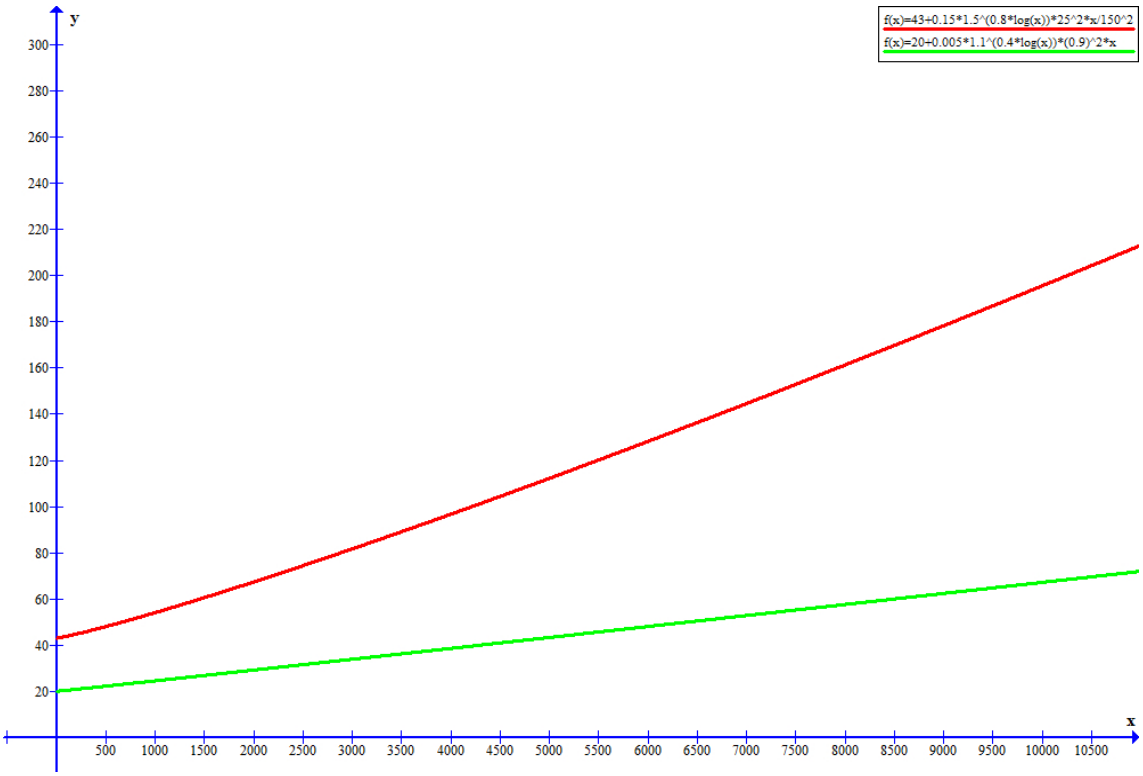
Uvedená rovnice je závislá především na minimální a maximálním poloměru. Je možné ji použít za podmínky, že počet hran není příliš vysoký, rozdíl maximálního a minimálního poloměru také není příliš vysoký a minimální poloměr je vetší než nula. První část tvoří základ prahu. Je nastavena jako přibližně třetina obvodu kruhu s minimálním poloměrem. Druhá část bere v úvahu neurčitost poloměru a tím i vyšší šance na vznik průsečíku. V případě největšího rozdílu poloměrů bude tato část rovna minimálnímu poloměru. Po přičtení k první části se práh rovná polovině obvodu nejmenšího kruhu. Tato hodnota ponechává dostatek prostoru pro neúplné nebo zkreslené kruhy. Je třeba podotknout, že v případě vyššího počtu hran ponechává funkce 11 větší počet falešných detekcí než funkce 10.

Celá tato metoda je sice rychlejší než předchozí, ale ponechává více falešných detekcí především pokud jsou v prohledávané oblasti shluky hran a mnoho volného prostoru okolo.

Nejlépe se osvědčila kombinace obou uvedených přístupů. Změnou prahové hodnoty pro jednotlivé oblasti se sníží počet možných kruhů detekovaných v první fázi. Ve druhé části algoritmu je vhodné vypočíst hodnotu prahu znovu. Pro hrubé vyhledávání na fotografiích jsem zvolil funkci 12. Pro potvrzovací část pak byla vybrána funkce ve tvaru 13. Obě dvě funkce jsou pouze dosazením do obecné formy 10. Pro snímky z videozáznamu pak zůstává funkce 11.

$$y = 43 + 0,15 \cdot 1,5^{0,8 \cdot log_{10}(h)} \cdot minr^2 \cdot h/(maxr^2) \tag{12}$$

$$y = 20 + 0,005 \cdot 1,1^{0,4 \cdot log_{10}(h)} \cdot (0,9 \cdot r)^2 \cdot h/((r+10)^2) \tag{13}$$



Obrázek 5: Funkce pro určení prahu akumulátoru

Transformace v druhé fázi se provádí nad čtvercovou oblastí o rozměrech $r + 10$, kde r je poloměr kruhu nalezeného v předchozím kroku. Přidání 10 pixelů umožní zpracovat i mírné natočení značky mimo oblast označenou v prvním fázi. Protože už existuje alespoň přibližná představa o rozměrech hledaného kruhu, je minimální poloměr možno vyjádřit jako násobek poloměru dříve nalezeného kruhu. V ideálním případě by měly být dokonce totožné, ale nižší rozlišení akumulátoru během předchozí Houghovy transformace může mít vliv na přesnost stejně jako okolí, které se vypouští až během druhé fáze.

Na obrázku 5 je zachycen průběh obou funkcí s parametry použitými v mém programu. Je patrné, že při porovnání s funkcí 12 je funkce 13 nejen posunutá ve směru osy y k počátku, ale i pomaleji roste. Jak již bylo zmíněno dříve, je tomu tak především kvůli mnohem menším vlivům okolí na výsledek transformací a tím i menším hodnotám v akumulátoru.

4.2 Binarizace

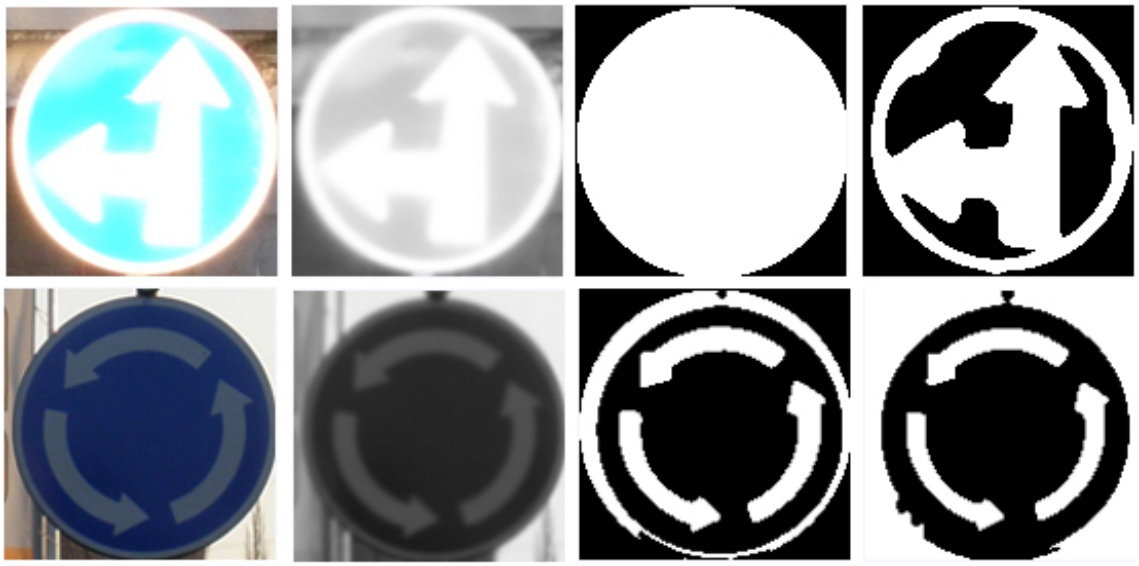
Tato část programu má rozhodující vliv na výpočet momentových charakteristik. Slouží především k tomu, aby se vyrovnala s rozdíly jasu vzniklými v důsledku změny barvy působením povětrnostních podmínek a různého osvětlení. Řešením je převést obraz do binární podoby tak, aby bylo jasně rozlišeno pole a symbol s lemem. Jak již bylo dříve zmíněno v kapitole 2, tyto dvě množiny bodů jsou si ve smyslu významu informace ekvivalentní.

Ve článcích [1] a [2] se navrhuje provést prahování s pevnou hodnotou pro všechny značky. Tu je třeba zjistit experimentálně. Tento přístup však neodráží různé vnější podmínky, jako je například osvětlení, a tedy i kontrast mezi polem a symbolem na jednotlivých značkách. Z tohoto důvodu je vhodné vypočítat práh pro každou nalezenou značku zvlášť.

4.2.1 Implementace

Nejdříve jsem vyzkoušel metodu Otsu[3]. Podstata spočívá v rozdělení bodů do dvou skupin podle jejich jasu tak, aby mezi těmito třídami byl co největší mezitřídní rozptyl ω_b, vyjádřený jako 14.

$$\begin{aligned} \omega_b^2 &= \omega_1 \cdot \omega_2 \cdot (\mu_1 - \mu_2)^2 \\ \omega_1^2 &= \sum_{i=0}^{T} \frac{(i-\mu_1)^2 \cdot \frac{n_i}{n}}{\frac{m_1}{n}} \\ \omega_2^2 &= \sum_{i=T}^{255} \frac{(i-\mu_2)^2 \cdot \frac{n_i}{n}}{\frac{m_2}{n}} \\ \mu_1 &= \sum_{i=0}^{T} i \cdot n_i \end{aligned} \tag{14}$$



Obrázek 6: Výsledky získané pomocí algoritmu Otsu. Zleva: původní barevný obraz, obraz převedený do stupňů šedi, prahování s oblastmi mimo kruh vyplněnými nulou, prahování s oblastmi mimo kruh vyplněnými průměrem

$$\mu_2 = \sum_{i=T}^{255} i \cdot n_i$$

V uvedených rovnicích n_i je počet pixelů s jasem i. Celkový počet obrazových bodů je označen jako n. Počet bodů s jasem menším nebo rovným prahu T je označen jako m_1, m_2 je počet pixelů s jasem vyšším než práh T. Aby bylo možno nalézt minimum výrazu 14 je nutné celý výpočet provést pro všechny možné prahy T.

Protože prostředky OpenCv neumožňují pracovat s kruhovými oblastmi, vznikala při binarizaci chyba, která byla způsobena pixely mimo značku. Tato odchylka nastává především pokud má celá značka výrazně odlišný jas od svého okolí. Proto selhávaly pokusy vyplnit okolí tmavou barvou. Pokusil jsem se tedy vyplnit body mimo kruh postupně minimem, maximem a průměrem z hodnot uvnitř značky. Bohužel minimum ani maximum neovlivnily mez natolik, aby byl problém odstraněn. Tyto statistické hodnoty jsou totiž vysoce závislé na přesnosti výběru značky a tedy i snadno ovlivnitelné hodnotou jasu mimo značku. Průměr umístil hodnotu jasu pixelů mimo značku do okolí prahu a zprahovaný obraz uvnitř značky odpovídal požadavkům. Nicméně hodnota jasu bodů mimo značku se nacházela v některých případech pod prahem, jindy naopak byla vyšší než práh. Ukázky výsledků těchto prvních pokusů jsou na obrázku 6.

Jako nejjednodušší řešení se ukázal převod nalezené značky do polárních souřadnic. Tím došlo k transformaci kruhové značky na čtvercovou. Práh vypočítaný nad takovýmto obrazem není ovlivňován jasem bodů mimo značku a odpovídá tedy požadavkům. Výpočet momentových charakteristik však probíhá v kartézském souřadném systému. Důvody budou uvedeny v kapitole 4.3.2. Aby tedy došlo k vyloučení bodů mimo značku z výběru

Obrázek 7: Výsledky získané pomocí algoritmu Otsu v polárních souřádnicích

je třeba je nastavit jejich hodnotu na hodnotu nižší nebo rovnu spočtenému prahu. Pro všechny možné prahy je takovou hodnotou nula. Druhou možností je vyplnit nulou tyto pixely až po prahování. Oba postupy dávají stejný výsledek.

Přestože došlo ke snížení vlivu okolí, jak je vidět na obrázku 7, může být tento vliv v případě nepřesného výběru pořád významný.

Modrá barva použitá jako pole u českých dopravních značek má výrazně vyšší odrazivost než červená používaná na lemu. Při snímcích získaných v šeru za předpokladu, že byla značka osvícena zdrojem světla, tedy může zpracovaný obraz být inverzní k obrazu získanému za běžných světelných podmínek a oproti vzorům. Uvedený problém se týká především značek Zákaz zastavení a Zákaz stání.

Další problémy vznikají u značek Zákaz vjezdu vozidel přepravujících nebezpečný náklad a Zákaz vjezdu vozidel přepravujících náklad, který může znečistit vodu. Barvy použité na symbolu vozidla mají nižší odolnost vůči povětrnostním podmínkám a rychleji vyblednou. Jejich vzory se pak začnou blížit vzorům jiných značek.

Proto jsem se rozhodl použít k binarizaci vlastní metodu. Ta však není založená pouze na jasové složce, ale na jednotlivých barevných složkách. Ty jsou sice závislé na osvětlení, ale lze je normalizovat podle osvětlení. Barevná informace přidává k jasové složce další

rozměry, ve kterých lze hledat podobnosti mezi jednotlivými oblastmi. Zároveň dovoluje převádět obraz do binární podoby podle barev používaných na dopravních značkách. Nutnou podmínkou je, aby byl po provedení všech kroků zvýšen rozdíl mezi symbolem a polem, respektive polem a lemováním. Celý postup má dva kroky. Prvním je normalizace barev. Při této úpravě barev jsem vyšel z článku [9]. Závěrečným krokem je pak vlastní převod takto pozměněného obrazu do binární podoby.

Protože je třeba brát v úvahu různé osvětlení v různých částech snímku, provádí se normalizace vždy nad čtvercovou oblastí o rozměrech $2r$ a se středem ve středu značky. Vstupem není pouze barevný obraz i jasová složka. Ta slouží k nastavení prahu pro jednotlivé barevné složky. Každá z barevných složek má vlastní prahovací funkci. Nejvýraznější změnou oproti článku [9] je nahrazení skokové funkce prahu funkcí spojitou. Porovnání průběhu obou funkcí je na obrázku 8.

Při použití skokové funkce docházelo v okolí bodů nespojitosti často ke špatnému prahování jednotlivých složek a tím i ke špatné normalizaci. Velmi častá také byla chyba při prahování obrazů se střední hodnotou jasu v rozmezí od osmdesáti do stoosmdesáti. Původní jednoduchá závislost se ukázala jako nevyhovující. Z těchto důvodů byla prahovací funkce nahrazena. Požadavkem bylo, aby byla funkce spojitá a aby nejrychleji rostla ve střední části oboru hodnot. Po sérii experimentů byla zvolena funkce jejíž obecný tvar je následující:

$$T(x) = \frac{255}{1 + e^{-\lambda(x - x_t)}} \tag{15}$$

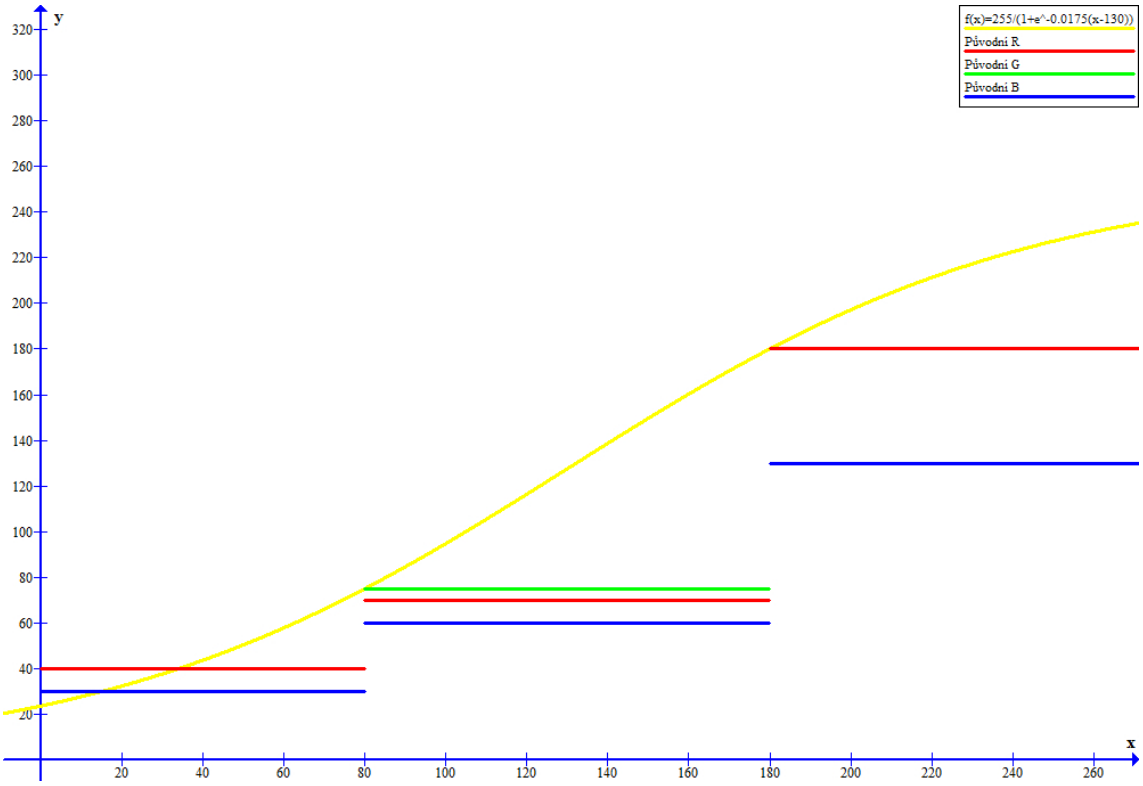
Proměnnou x je průměr jasu ve zpracovávané oblasti. Parametr λ ovlivňuje strmost této funkce a tím rychlost růstu prahu. Zbývající parametr x_t je posunem v kladném směru osy průměru jasu. Manipulací s jednotlivými parametry lze jednoduše upravit tvar prahovací funkce pro jednotlivé kanály. Pro mé účely jsem zvolil $\lambda = 0.0175$ a $x_t = 130$ pro všechny barevné kanály.

Z grafu na obrázku 8 je patrné, že uvedená funkce nejen roste rychleji, ale dokonce jsou i její hodnoty mnohem vyšší. Přesto se ukázalo, že je daleko vhodnější a pro mé učely postačující.

Po provedení těchto úprav by mělo dojít ke snížení počtu možných barev ve zpracovávané oblasti na pouhých osm. Tento počet umožňuje definovat několik pravidel pro převod takovéhoto obrazu do binární podoby. Před aplikací jednotlivých pravidel je třeba spočíst relativní četnosti v jednotlivých barevných kanálech. Pravidla se pak probírají postupně a po nalezení prvního vyhovujícího se zpracovává další bod.

Pro každý převáděný bod potom platí:

1. Pixel obsahující všechny tři složky se vždy převádí na hodnotu 1
2. Pokud převažuje modrá nad zelenou a červenou platí následující pravidla:
   (a) Pokud jsou zelená a modrá přibližně v rovnováze a zelená převažuje nad červenou platí tato pravidla:
      i. Obsahuje-li zpracovávaný bod zelenou a červenou složku nebo jen zelenou, převede se na 1



Obrázek 8: Funkce použitá pro výpočet prahu pro jednotlivé barevné kanály

   ii. Obsahuje-li pixel pouze zelenou a modrou složku a červená barva není na značce významně zastoupena, převede se na 0
   iii. Obsahuje-li pixel pouze zelenou a modrou složku a červená barva je na značce významně zastoupena, převede se na 255
   iv. Ostatní se převádí na 0

(b) Pokud převažuje zelená nad červenou:
   i. Obsahuje-li zpracovávaný bod modrou složku, červená složka je nulová a červená barva není významně zastoupena, převede se na 0
   ii. Obsahuje-li zpracovávaný bod červenou složku, ostatní složky nejsou v bodě přítomny zároveň a červená barva je významně zastoupena, převede se na 0
   iii. Vše ostatní se převede na 1

(c) Pokud převažuje červená nad zelenou
   i. Pokud zpracovávaný pixel obsahuje modrou složku nebo červenou složku a zároveň červená není v obraze významněji zastoupena, převede se na 0
   ii. Pokud zpracovávaný pixel obsahuje červenou složku, ostatní složky nejsou obě dvě zároveň v bodě obsaženy a červená je v obraze významně zastoupena, převede se na 0
   iii. Ostatní se převede na 1

(d) V ostatních případech
   i. Obsahuje-li zpracovávaný bod červenou složku a buď zelená nebo modrá složka jsou nulové, případně obě dvě, převede se na 0
   ii. Ostatní se převede na 1

3. Pokud převažuje zelená nad modrou a červenou platí následující pravidla:

(a) Pokud nejsou zelená a modrá přibližně v rovnováze:
   i. Pokud pixel obsahuje červenou složku převede se na 0
   ii. Ostatní se převádí na 1

4. Pokud převažuje červená nad modrou a zelenou platí následující pravidla:

(a) Pokud pixel neobsahuje žádnou ze složek převádí se na 0

(b) Pokud pixel neobsahuje modrou nebo zelenou, případně obě dvě, převede se na 0

(c) Ostatní se převádí na 1

5. Pixel neobsahující žádnou složku se převede 0

6. Pokud není výše uvedeno jinak převede se pixel na 0

Pravidlo 1 zvýrazňuje symboly u příkazových značek a bílé pole, případně symbol, u zákazových značek. Samo o sobě není nijak zvláštní. Pravidlo 5 pak zachovává černou barvu. Pravidlo 6 je výchozím doplněním k ostatním pravidlům.

Pravidla bodu 2 se týkají příkazových značek. Pokud jsou modrá se zelenou v rovnováze, došlo ke standartizaci pole značky na azurovou. Tuto situaci popisuje bod (a). Obsahuje-li zpracovávaný bod pouze zelenou složku, nebo zelenou a červenou složku, neodpovídá takováto barva určitě barvě pole příkazové značky a převádí se podle pravidla i se na 1. Pokud zpracovávaný bod obsahuje pouze zelenou a modrou složku je tato barva určitě barvou pole. Pokud je tato značka příkazová, červená barva na ní určitě nemá výraznější zastoupení a bod se tedy převede podle pravidla ii na 0. Pokud je výskyt červené barvy na celé ploše značky významný, je tato značka zákazová a bod se převede na 1, jak říká pravidlo iii. Vše ostatní se pak převede na 0.

Pokud se jen část modrého pole převedla na azurovou a zbytek zůstal modrý, nastává situace podlu bodu (b). V případě, že není červená ve značce významněji zastoupena, jedná se o příkazovou značku a tedy vše, co obsahuje modrou složku a zároveň neobsahuje červenou složku, se převede podle pravidla i na 0. Pravidlo ii říká, že pokud je červená ve značce zastoupena významně, jedná se o zákazovou značku a na 0 se tedy převádí vše co obsahuje červenou složku, pokud neobsahuje zároveň všechny ostatní. Pravidlo iii převádí vše ostatní na 1.

Pravidla bodu (c) jsou obdobou bodu (b). Bod (d) předpokládá, že značka je zákazová. Všechno co obsahuje červenou složku se tedy bodle pravidla i převede na 0. Vše ostatní se pak převede na 1.

Pravidla v bodu 3 se mohou uplatnit především u příkazových značek. Vlivem osvětlení se může stát, že barva jejich pole bude normalizována na azurovou. V takové situaci musí převládat zelená, neboť s velmi vysokou pravděpodobností bude obsažena v každém bodu náležejícím značce. V takovéto situaci by měly být zelená a modrá složka v rovnováze. Prahuje se pak podle pravidel 1 a 6. Pokud tomu tak není uplatní se pravidla bodu 3.a. Tato podmínka naznačuje, že značka je zákazová a má červený lem. Pravidlo i potlačí lem a pravidlo ii zvýrazní symbol.

Pravidla bodu 4 se týkají především zákazových značek. U těchto značek totiž musí červená převažovat už jen kvůli červenému lemování. Bod (a) potlačí černou barvu. Obdobně pracuje bod (b) s červenou, purporovou a žlutou. Zbývá už jen situace, kdy pixel obsahuje modrou i zelenou současně, může se jednat buď o bílou nebo azurovou. Obě dvě barvy v případě zákazových značek tvoří pole a proto patří do výběru.

Body mimo kruh je opět nutné vyplnit nulovými hodnotami, aby neovlivňovaly další výpočty.

Výsledky této metody jsou v případě odlesků od dopravních značek za běžného osvětlení lepší, než při prahování pouze podle jasové funkce. Jak je patrné z obrázku 9, tato metoda odstraňuje vlivy nejen okolí ale i odlesků. Dále pak přináší stejné výsledky pro značky používající červené lemování a modré pole zároveň za různých podmínek osvětlení.

Obrázek 9: Ukázka výstupů při použití pravidel 1-5

4.3 Výpočet momentových charakteristik

4.3.1 Momety a momentové invarianty

Momentové charakteristiky v analýze obrazu jsou v podstatě převzaty ze statistiky, kde popisují rozložení hodnot náhodné veličiny v prostoru proměnných této veličiny. Pro spojité náhodné veličiny je moment m řádu i definován jako:

$$m_i = \int_{\infty}^{-\infty} x^i \cdot f(x)dx \tag{16}$$

Pro diskrétní náhodné veličiny se nahradí určitý integrál sumou. Vztah má potom následující tvar:

$$m_i = \sum_x x^i \cdot f(x) \tag{17}$$

Na každý zpracovávaný obraz se lze dívat jako na diskrétní náhodnou veličinu proměnných x a y. Hodnota náhodné veličiny je pak dána jasem na souřadnicích (x, y). Protože jednotlivé momenty popisují rozložení náhodné veličiny v prostoru, v tomto případě jasu, popisují i jednotlivé obrazy. Obrazy s rozdílným rozložením jasu jsou popsány odlišnou hodnotou jasu a je tedy možné je pomocí momentů od sebe odlišit.

Klasické momenty jsou ovšem závislé na poloze rozpoznávaného objektu v obraze. Je tedy třeba transformovat souřadnou soustavu, jejíž souřadnice tvoří proměnné náhodné veličiny jasu, na souřadnou soustavu se středem v těžišti rozpoznávaného objektu. Pro transformaci souřadné soustavy x, y souřadnou soustavu $\hat{x}, \hat{y}$ se středem v těžišti rozpoznávaného objektu platí následující vztahy:

$$m_{i,j} = \sum_x \sum_y x^i \cdot y^j \cdot f(x, y) \tag{18}$$

$$x_T = \frac{m_{1,0}}{m_{0,0}} \tag{19}$$

$$y_T = \frac{m_{0,1}}{m_{0,0}} \tag{20}$$

$$\hat{x} = x - x_T \tag{21}$$

$$\hat{y} = y - y_T \tag{22}$$

V uvedených rovnicích x_T a y_T označují těžiště a střed nové souřadné soustavy. Pokud výrazy 19 a 20 rozepíšeme na 23 a 24 a podíváme-li se na ně pozorněji, uvidíme že se jedná o vzorec pro výpočet těžiště plochy, která má konečný počet bodů.

$$x_T = \frac{\sum_x \sum_y x \cdot f(x, y)}{\sum_x \sum_y f(x, y)} \tag{23}$$

$$y_T = \frac{\sum_x \sum_y y \cdot f(x, y)}{\sum_x \sum_y f(x, y)} \tag{24}$$

$$x_T = \frac{x_1 + x_2 + \ldots + x_n}{n} \tag{25}$$

Ve jmenovateli výrazu 23 je součet všech jasů v obraze. Čitatel je součtem všech souřadnic x náležejících objektu násobený jasem příslušných pixelů. Předpokládejme že vstupní obraz je binární. V takovém případě je ve jmenovateli pouze součet souřadnic x všech bodů náležejících objektu a ve jmenovateli počet všech bodů. Tento výsledek přesně odpovídá vzorci 25. Tato rovnice odpovídá výpočtu těžiště plochy se stejnou hustotou ve všech bodech, přičemž daná plocha má konečný počet bodů. Pokud budeme uvažovat nebinarizovaný obraz, situace odpovídá ploše s rozdílným rozložením hustoty.

Momenty vypočtené ze souřadnic $\hat{x}$ a $\hat{y}$ se nazývají centrální momenty μ. Tyto momenty jsou sice nezávislé na poloze rozpoznávaného objektu v obraze, ale jsou závislé na ploše objektu. Pokud se mají rozpoznávat objekty stejného tvaru, ale různé velikosti, je nutné tyto momenty vztáhnout vůči ploše rozpoznávaného objektu. Takovéto momenty se nazývají normalizované momenty. Vypočítají se podle následujících vzorců:

$$\eta_{i,j} = \frac{\mu_{i,j}}{mu_{0,0}^{\gamma}} \tag{26}$$

$$\gamma = \frac{i+j}{2} + 1 \tag{27}$$

Normalizované momenty jsou však stále závislé na rotaci objektu. Řešením je použití momentových invariantů. Dnes nejrozšířenější jsou Huovy invarianty [4] odvozené už v roce 1962. Tato sada momentových invariantů má následující tvar:

$$hu_1 = \eta_{20} + \eta_{02} \tag{28}$$

$$hu_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{29}$$

$$hu_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{30}$$

$$hu_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{31}$$

$$\begin{aligned} hu_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{32}$$

$$hu_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{33}$$

$$\begin{aligned} hu_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] - \\ & - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{34}$$

J. Flusser a T. Suk ve své práci [5] při odvozování pravidel pro tvorbu momentových invariantů odvozených z libovolných řádů momentů dokázali, že invariant hu_3 je kombinací invariantů hu_5 a hu_7. Proto není pro rozpoznávání objektů podstatný a lze ho vypustit.

Tito autoři v jedné ze svých předchozích prací [6] vytvořili čtyři následující momentové invarianty postačující k rozpoznávání obrazu:

$$\pi_1 = \mu_{0,0}^{-4}(\mu_{2,0}\mu_{0,2} - \mu_{1,1}^2) \tag{35}$$

$$\pi_2 = \mu_{0,0}^{-10}(\mu_{3,0}^2\mu_{0,3}^2 - 6\mu_{3,0}\mu_{2,1}\mu_{1,2}\mu_{0,3} + 4\mu_{3,0}\mu_{1,2}^3 + 4\mu_{0,3}\mu_{2,1}^3 - 3\mu_{2,1}^2\mu_{1,2}^2) \tag{36}$$

$$\pi_3 = \mu_{0,0}^{-7}[\mu_{2,0}(\mu_{2,1}\mu_{0,3} - \mu_{1,2}^2) - \mu_{1,1}(\mu_{3,0}\mu_{0,3} - \mu_{2,1}\mu_{1,2}) + \mu_{0,2}(6\mu_{3,0}\mu_{1,2} - \mu_{2,1}^2)] \tag{37}$$

$$\begin{aligned}\pi_4 = \mu_{0,0}^{-11}[&\mu_{2,0}^3\mu_{0,3}^2 - 6\mu_{2,0}^2\mu_{1,1}\mu_{1,2}\mu_{0,3} - 6\mu_{2,0}^2\mu_{0,2}\mu_{2,1}\mu_{0,3} + 9\mu_{2,0}^2\mu_{0,2}\mu_{1,2}^2 + \\ &+12\mu_{2,0}\mu_{1,1}^2\mu_{2,1}\mu_{0,3} + 6\mu_{2,0}\mu_{1,1}\mu_{0,2}\mu_{0,3}\mu_{3,0} - 18\mu_{2,0}\mu_{1,1}\mu_{0,2}\mu_{2,1}\mu_{1,2} - \\ &-8\mu_{1,1}^3\mu_{0,3}\mu_{3,0} - 6\mu_{2,0}\mu_{0,2}^2\mu_{3,0}\mu_{1,2} + 9\mu_{2,0}\mu_{0,2}^2\mu_{1,2}^2 + 12\mu_{1,1}^2\mu_{0,2}\mu_{3,0}\mu_{1,2} - \\ &-6\mu_{1,1}\mu_{0,2}^2\mu_{3,0}\mu_{2,1} + \mu_{0,2}^3\mu_{3,0}^2]\end{aligned} \tag{38}$$

Výše uvedené invarianty jsou nezávislé na posuvu, rotaci a zvětšení.

4.3.2 Implementace a vzniklé problémy

Pro svou implementaci jsem jako rozpoznávací příznaky zvolil Huovy invarianty. Jejich výhodou je především to, že jsou obsaženy v knihovnách OpenCv. Výpočet těchto momentů je tedy plně v režii OpenCV.

Hodnoty jednotlivých invariantů pro některé dopravní značky se však pohybují velmi blízko nule. Navíc má každý z invariantů odlišný exponent. Pro další zpracování je vhodné přiblížit maximální hodnoty pro každý z invariantů co nejblíže jedné. Tím se zajistí, aby měl každý invariant stejný vliv na následné vyhodnocení. Vzhledem k tomu, že pro některé vzory se jeden vypočtený invariant pohyboval v řádech tisícin a pro jiné až v mnohem nižších řádech, není možné cíle dosáhnout pouze jednoduchým násobením. Tento problém byl efektivně vyřešen v části klasifikace.

Nyní nastává čas zamyslet se nad důvody výpočtu jednotlivých momentů v kartézské soustavě. Pokud by byl obraz ponechán v polárních souřadnicích, veškeré rotace by se projevily jako posuny. Vzhledem k periodicitě goniometrických funkcí je i tato soustava souřadnic periodická, stejně jako každý obraz v ní. Pokud by byl symbol umístěn tak,že některou jeho částí prochází hranice obrazu na ose úhlu, navazující část se objeví u protější hranice na její vnitřní straně. Takto vzniklý obraz je z hlediska Huových invariantů odlišný. Musely by tedy být definovány vzory pro všechna možná rozdělení symbolu.

Možným řešením je vytvořit obraz, ve kterém se bude zprahovaný obraz opakovat dvakrát. Poté by bylo nutné vybrat takovou část o rozměrech zprahovaného obrazu, aby se v ní vyskytoval symbol celý. Z těchto důvodů je vhodnější použít kartézskou souřadnou soustavu.

Porovnáním jednotlivých vzorů českých dopravních značek ovšem vyvstává problém. Nejen že nelze jen s pomocí momentových invariantů rozlišit přikázaný směr jízdy vlevo, přikázaný směr jízdy vpravo a přikázaný směr jízdy rovně, ale dokonce tyto značky nelze rozlišit ani od přikázaného směru objíždění vlevo a přikázaného směru objíždění vpravo. Tato situace je dána faktem, že momentové invarianty jsou nezávislé na rotaci a u uvedených značek je rotace symbolu jediným rozlišujícím znakem.

Pro tyto dopravní značky je tedy nutné zavést ještě příznak natočení vůčí hlavní ose rozpoznávaného objektu. Úhel natočení lze vypočíst podle následujícího vzorce:

$$tg2\Theta = \frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}} \tag{39}$$

$$\Theta = \frac{1}{2} arctg(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}) \tag{40}$$

Tento příznak by měl přijít na řadu až pokud je zjištěno, že je opravdu potřebný. Tím se sníží výpočetní složitost nejen pro výpočet charakteristik objektů, ale i pro následné vyhodnocení příznaků.

4.4 Klasifikace

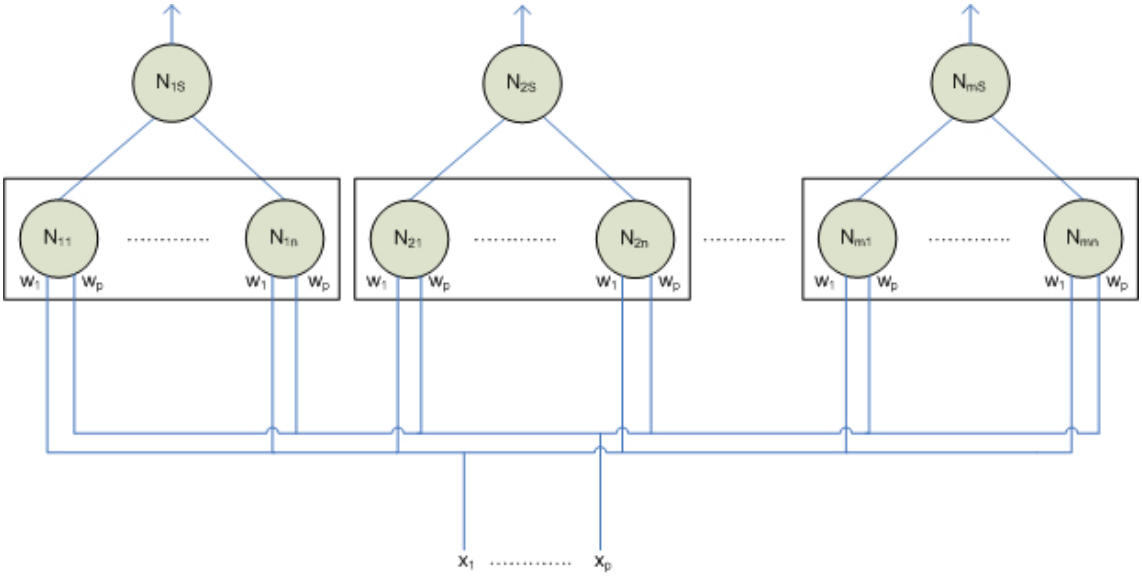
V článku [1] byla pro účel klasifikace použita pravděpodobnostní neuronová síť. Schéma takovéto sítě je na obrázku 10. Výhodou je rychlý proces učení. Vzory se uloží do vah v první vrstvě neuronů a tím učení sítě skončí. Potenciál v první vrstvě neuronů se spočte podle rovnice 41. Aktivační funkce pro tuto vrstvu má tvar 42. Druhá vrstva této sítě je sumační. Všechny váhy jsou v ní rovny $\frac{1}{\sqrt{2\pi\sigma^2}}$. Aktivační funkcí druhé vrstvy je identita. Vstup pak náleží třídě, jejíž výstup dává nejvyšší hodnotu. Poněkud problematické je zjištění parametru σ. Ten je třeba vypočítat jako střední hodnotu rozptylu vzorů v jednotlivých třídách.

$$d = \sum_i (w_i - x_i)^2 \tag{41}$$

$$y = e^{-\frac{d}{2\sigma^2}} \tag{42}$$

Jak je vidět z uvedených rovnic, výstupní hodnota hodnota je závislá především na vzdálenosti d. Vzorů musí být pro každou třídu tolik, aby dokázaly pokrýt celý prostor, ve kterém se vyskytuje daná třída.

Naproti tomu vícevrstvá neuronová síť dokáže pokrýt celý prostor třídy vzorů s takovým počtem vzorů, který dostatečně přesně definuje hranice této třídy. Taková síť se sice pomaleji učí, ale ve výsledku může být rychlejší než vyhodnocování vzdálenosti vůči velkému počtu vzorů. Z tohoto důvodu jsem nejprve vyzkoušel vícevrstvou neuronovou síť.



Obrázek 10: Pravděpodobnostní neuronová síť

Bohužel implementovaná síť se nebyla schopna spolehlivě naučit rozpoznávat jednotlivé vzory. Musel jsem se tedy vrátit o krok zpět a vyjít z pravděpodobnostní sítě. Jak již bylo řečeno dříve, hodnota výstupu závisí především na vzdálenosti d vektoru příznaků od vektoru vzorů. Zcela ekvivalentím řešením je tedy, hledat minimální vzdálenost d. Takováto neuronová síť se pak řadí mezi kompetitivní neuronové sítě.

4.4.1 Vícevrstvá neuronová síť a backpropagation

Vícevrstvé neuronové sítě se skládají z několika vrstev neuronů. Každý neuron má svůj vektor vah w, potenciál z, který je funkcí vstupu a vah a svou aktivační funkci, která transformuje potenciál z na výstup y.

Potenciál neuronu se spočte jako součet vstupu x vážený vektorem vah w.

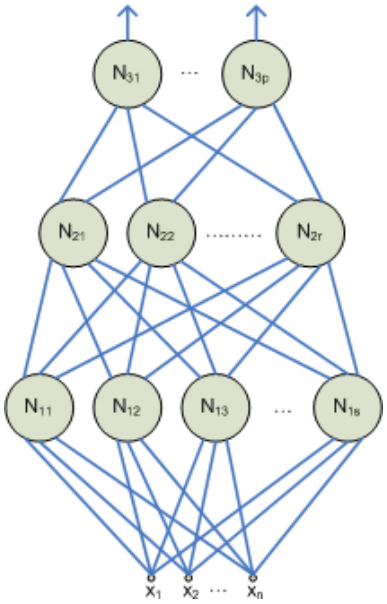
$$z = \sum_i w_i \cdot x_i \tag{43}$$

Jako aktivační funkce se nejčastěji volí sigmoida. Tvar této funkce je následující:

$$y = \frac{1}{1 + e^{-\lambda(z-\theta)}} \tag{44}$$

Parametr λ určuje strmost aktivační funkce, často se volí $\lambda = 1$. Práh θ je hodnota, která musí být překročena, aby došlo k excitaci neuronu

Na obrázku 11 je schéma třívrstvé neuronové sítě. Vstupy jsou přivedeny do spodní vrstvy neuronů. Každý neuron této vrstvy má tolik vah, kolik je vstupů do celé sítě. Druhá vrstva neuronů zpracovává výstupní signál z první vrstvy. Všechny výstupy první vrstvy



Obrázek 11: Vícevrstvá neuronová síť

tvoří vstupy pro druhou vrstvu. V každém z neuronů druhé vrstvy je počet vah roven počtu neuronů první vrstvy. Třetí vrstva je obdobou vrstvy druhé. Výstupy z ní však tvoří výstup z celé sítě. Při řešení klasifikačních úkolů je tedy třeba vybrat neuron s maximální hodnotou výstupu.

Váhy se nastavují během procesu učení. Pro vícevrstvé neuronové sítě se dnes běžně používá algoritmus backpropagation. Na počátku se váhy všech neuronů nastaví na náhodné hodnoty. Po vložení vzoru itého se spočte výstup y_i a z něj chyba E oproti požadovanému výsledku o_i.

$$E = \frac{1}{2} \sum_i \sum_j (y_{ij} - o_{ij})^2 \tag{45}$$

Účelem učení sítě je nastavit váhy tak, aby chyba E byla co nejnižší. Z rovnice 45 je patrné, že chyba na výstupu je dána výstupem y. Ten je určen pomocí potenciálu z, který je dán vstupy x a váhami w. Matematický zapis je tedy následující:

$$E(f(y(z(x, w)))) \tag{46}$$

Při hledání minima funkce 46 pro každý neuron algoritmus backpropagation využívá gradientní metodu. Uvedený výraz je tedy třeba zderivivat podle jednotlivýchch vah.

$$\frac{\delta E}{\delta w_i} = \frac{\delta E}{\delta y} \cdot \frac{\delta y}{\delta z} \cdot \frac{\delta z}{\delta w_i} \tag{47}$$

Protože se jedná o složenou funkci, skládá se derivace ze tří dílčích derivací. Nejjednodušší je situace v případě potenciálu $z(x, w)$. Derivací získáváme výraz 49.

$$z = \sum_i w_i \cdot x_i \tag{48}$$

$$\frac{\delta z}{\delta w_i} = x_i \tag{49}$$

Pro jednoduchost nyní uvažujme, že práh θ v aktivační funkci je roven nule. Derivací podle potenciálu z a postupnými úpravami lze dojít k rovnici 56.

$$y = \frac{1}{1 + e^{-\lambda z}} \tag{50}$$

$$\frac{\delta y}{\delta z} = -(1 + e^{-\lambda z})^{-2} \cdot (0 - \lambda e^{-\lambda z}) \tag{51}$$

$$\frac{\delta y}{\delta z} = \lambda \frac{e^{-\lambda z}}{1 + e^{-\lambda z}} \cdot \frac{1}{1 + e^{-\lambda z}} \tag{52}$$

$$\frac{\delta y}{\delta z} = \lambda y \frac{e^{-\lambda z}}{1 + e^{-\lambda z}} \tag{53}$$

$$\frac{\delta y}{\delta z} = \lambda y \frac{e^{-\lambda z} + 1 - 1}{1 + e^{-\lambda z}} \tag{54}$$

$$\frac{\delta y}{\delta z} = \lambda y (\frac{e^{-\lambda z} + 1}{1 + e^{-\lambda z}} - \frac{1}{1 + e^{-\lambda z}}) \tag{55}$$

$$\frac{\delta y}{\delta z} = \lambda y (1 - y) \tag{56}$$

Poslední částí rovnice 47 je derivace chyby podle výstupu neuronu. Pro neurony ve výstupní vrstvě je řešení jednoduché. Chyba E je dána součtem rozdílů aktuálního výstupu vůči požadovanému. Derivace pak vypadá následovně:

$$\frac{\delta E}{\delta y} = \frac{\frac{1}{2}(y - o)^2}{\delta y} \tag{57}$$

$$\frac{\delta E}{\delta y} = y - o \tag{58}$$

Poněkud obtížnější je situace v nižších vrstvách. Neurony se sice podílejí na chybě na výstupu, ale ne každý se na této chybě podílí stejnou měrou. Vezměme v úvahu neurony předposlední vrstvy. Jejich výstup je vstupem pro poslední vrstvu neuronů. Chybný výstup z předposlední vrstvy způsobuje chybnou excitaci v poslední vrstvě. Celou závislost pak lze zapsat následujícím způsobem:

$$E(E^i(z^i(y))) \tag{59}$$

Derivace pak vypadá:

$$\frac{\delta E}{\delta y} = \sum_i \frac{\delta E^i}{\delta z^i} \cdot \frac{\delta z^i}{\delta y} \tag{60}$$

Suma se provádí přes všechny neurony vyšší vrstvy k vrstvě aktuální. Proměnná E^i označuje chybu itého neuronu vyšší vrstvy. Potenciál z^i je také potenciálem itého neuronu vyšší vrstvy. Z těchto skutečností vyplývá, že je třeba znát údaje z vyšší vrstvy. Protože je mezi nimi i potenciál, není možné změnit váhy ihned po spočtení změny vah pro jeden neuron, ale až poté co jsou spočteny změny vah pro všechny neurony v síti. Druhá část derivovaného výrazu je v podstatě obdobou vzorce 49, rozdíl je pouze v symbolech a v proměnné, podle které se derivuje. Lze tedy provést substituci a výsledek pak vypadá:

$$\frac{\delta E}{\delta y} = \sum_i \frac{\delta E^i}{\delta z^i} \cdot w^i \tag{61}$$

Výsledný vzorec pro změnu vah je pak následující:

$$w_i(t+1) = w_i(t) + \eta \frac{\delta E}{\delta y} \cdot \lambda y(1-y) \cdot x_i \tag{62}$$

Koeficient učení η ovlivňuje velikost kroků, o které se váhy mění. V případě, že bude příliš vysoký, může učení trvat dlouho, stejně jako v případě příliš nízkého koeficientu učení. Běžně se volí η přibližně 0,3.

Pro urychlení nalezení minima chybové funkce E je možné měnit podle potřeby i strmost aktivační funkce λ a její práh θ. Změna těchto hodnot umožňuje individuální nastavení pro každý neuron a tím lepší adaptaci celé neuronové sítě. Chybová funkce E je tedy funkcí nejen vah w, ale i strmosti λ a prahu θ.

$$E(y(z(x), \lambda, \theta)) \tag{63}$$

Je tedy nutné hledat nejen nastavení jednotlivých vah ale i parametrů λ a θ. Jejich optimální nastavení se opět hledá pomocí gradientní metody. Vzorce pro jejich úpravu jsou tedy následující:

$$\lambda(t+1) = \lambda(t) - \xi \frac{\delta E}{\delta \lambda} \tag{64}$$

$$\lambda(t+1) = \theta(t) - \psi \frac{\delta E}{\delta \theta} \tag{65}$$

Opět je tedy nutno derivovat chybovou funkci E podle nastavovaného parametru. Výraz $\frac{\delta E}{\delta y}$ v následující rovnici se vypočte stejným způsobem jako při úpravě vah.

$$\frac{\delta E}{\delta \lambda} = \frac{\delta E}{\delta y} \cdot \frac{\delta y}{\delta \lambda} \tag{66}$$

$$\frac{\delta y}{\delta \lambda} = (1 + e^{-\lambda(z-\theta)})^{-2} \cdot (0 - \frac{\delta(1 + e^{-\lambda(z-\theta)})}{\delta \lambda}) \tag{67}$$

$$\frac{\delta y}{\delta \lambda} = (1 + e^{-\lambda(z-\theta)})^{-2} \cdot (-e^{-\lambda(z-\theta)})(z-\theta)(-\lambda)) \tag{68}$$

$$\frac{\delta y}{\delta \lambda} = \lambda(z-\theta) \cdot \frac{1}{(1 + e^{-\lambda(z-\theta)})} \cdot \frac{e^{-\lambda(z-\theta)}}{1 + e^{-\lambda(z-\theta)}} \tag{69}$$

$$\frac{\delta y}{\delta \lambda} = \lambda y(z-\theta) \cdot \frac{(1 + e^{-\lambda(z-\theta)} - 1)}{1 + e^{-\lambda(z-\theta)}} \tag{70}$$

$$\frac{\delta y}{\delta \lambda} = \lambda y(z-\theta) \cdot (\frac{1 + e^{-\lambda(z-\theta)}}{1 + e^{-\lambda(z-\theta)}} - \frac{1}{1 + e^{-\lambda(z-\theta)}}) \tag{71}$$

$$\frac{\delta y}{\delta \lambda} = \lambda y(z-\theta)(1-y) \tag{72}$$

Po dosazení výsledku do vzorce 66 vypadá vztah pro úpravu strmosti následovně:

$$\frac{\delta E}{\delta \lambda} = \frac{\delta E}{\delta y} \cdot \lambda y(z-\theta)(1-y) \tag{73}$$

Obdobným způsobem se vyjádří výraz $\frac{\delta E}{\delta \theta}$.

$$\frac{\delta E}{\delta \theta} = \frac{\delta E}{\delta y} \cdot \frac{\delta y}{\delta \theta} \tag{74}$$

$$\frac{\delta y}{\delta \theta} = (1 + e^{-\lambda(z-\theta)})^{-2} \cdot (0 - \frac{\delta(1 + e^{-\lambda(z-\theta)})}{\delta \theta}) \tag{75}$$

$$\frac{\delta y}{\delta \theta} = (1 + e^{-\lambda(z-\theta)})^{-2} \cdot (0 - e^{-\lambda(z-\theta)}\lambda) \tag{76}$$

$$\frac{\delta y}{\delta \theta} = -\lambda \frac{1}{1 + e^{-\lambda(z-\theta)}} \cdot \frac{e^{-\lambda(z-\theta)}}{1 + e^{-\lambda(z-\theta)}} \tag{77}$$

$$\frac{\delta y}{\delta \theta} = -\lambda y \cdot \frac{1 + e^{-\lambda(z-\theta)} - 1}{1 + e^{-\lambda(z-\theta)}} \tag{78}$$

$$\frac{\delta y}{\delta \theta} = -\lambda y \cdot (\frac{1 + e^{-\lambda(z-\theta)} - 1}{1 + e^{-\lambda(z-\theta)}} - \frac{1}{1 + e^{-\lambda(z-\theta)}}) \tag{79}$$

$$\frac{\delta y}{\delta \theta} = -\lambda y(1-y) \tag{80}$$

Dosazením do rovnice 74 získáváme výsledný vzorec pro úpravu prahu:

$$\frac{\delta E}{\delta \theta} = -\frac{\delta E}{\delta y} \cdot \lambda y(1-y) \tag{81}$$

Výraz $\frac{\delta E}{\delta y}$ se opět spočte stejným způsobem jako při úpravě vah. Změna strmosti λ a prahu θ se opět provádí až po vypočtení této změny pro všechny neurony v síti.

Nevýhodou tohoto učení je, že ne vždycky nutně musí dojít k nalezení globálního minima funkce $E(w)$. Může se stát, že se algoritmus zastaví v některém z lokálních minim a nebude schopen váhy upravit lépe. V takovém případě je nutné začít znova od začátku s novým nastavením vah.

Počet vrstev takovéto sítě může být libovolný, ale pro rozpoznávání jakékoliv skupiny vzorů jsou dostatečné tři vrstvy. Přidáním dalších vrstvev dojde k prodloužení procesu učení i běžného výpočtu. Počet neuronů ve výstupní vrstvě je dán počtem rozpoznávaných vzorů. Počet neuronů v ostatních vrstvách se volí podle potřeby. Pokud je tento počet příliš nízký, nemusí být daná síť schopna naučit se rozpoznávat všechny požadované vzory. Při příliš vysokém počtu neuronů naopak síť vzory příliš konkretizuje a nemusí být shopna rozeznat podobné vzory, patřící do stejné skupiny.

Přestože knihovna OpenCv obsahuje neuronové sítě a algoritmy pro jejich učení, nejsou momentálně v podobě vhodné k použití. Některé metody, jako například uložení sítě do souboru, nejsou totiž stále implementovány. Z tohoto důvodu jsem se rozhodl použít vlastní implementaci vícevrstvé neuronové sítě a algoritmu backpropagation.

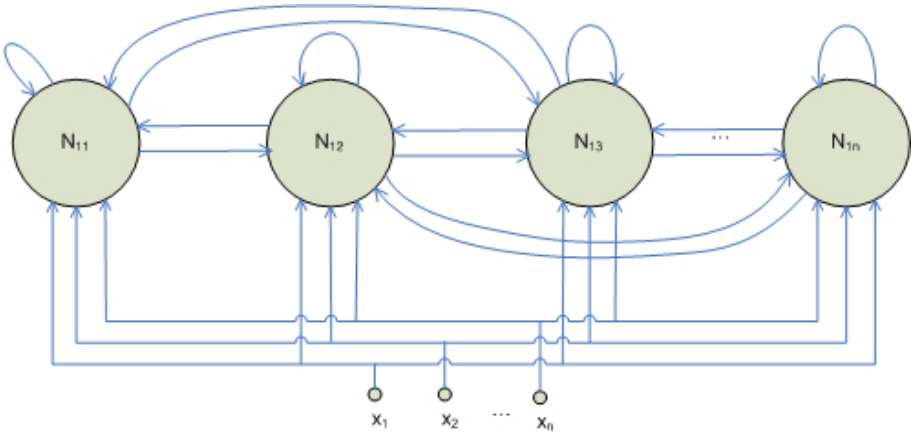
Nejprve jsem se pokusil rozlišit od sebe navzájem značky Zákaz vjezdu a Zákaz vjezdu do jednosměrné ulice. V prostoru vstupů jsou tyto dva vzory lineárně separovatelné a lze je tedy rozpoznávat i za použití jediného neuronu. Jako vstupy do neuronové sítě jsem použil Huovy invarianty v původní podobě. Výsledek odpovídal očekávání.

Dále jsem tedy vyzkoušel naučit síť 25 vybraných druhů značek. Připravil jsem síť o struktuře 100-50-25. Po velkém počtu iterací jsem učení zastavil a vyzkoušel přidat další neurony. Síť tedy získala rozměry 200-100-25. S invarianty v původní podobě se ale nebyla ani takto rozsáhlá síť schopna naučit vzory na nižší chybu než 18% na jeden vzor trénovací množiny.

Možným řešením by bylo zvýšení počtu neuronů ve vnitřních vrstvách sítě. Tím by ale došlo ke snížení rychlosti výpočtu. V porovnání s pravděpodobnostní sítí jeden neuron vstupní vrstvy odpovídá jednomu uloženému vzoru v pravděpodobnostní síti. Obrovský nárust však přichází ve druhé vrstvě. Jestliže má první vrstva n neuronů a druhá vrstva m neuronů, pak počet neuronů druhé vrstvy odpovídá $\frac{mn}{7}$ vzorům uloženým v pravděpodobnostní síti. Obdobný vztah pak platí i pro poslední vrstvu. Vzhledem k těmto počtů jsem se rozhodl přistoupit k použití obdoby pravděpodobnostní sítě, tedy přímému uložení vzorů.

Předtím jsem však vyzkoušel ještě druhou možnost. Tou je pokusit se upravit vstupy do podoby přijatelnější pro neuronovou síť. Vzhledem k tomu, že vzory jsou rozloženy téměř po celém intervalu nula až jedna, nepřipadá v úvahu pouze vynásobení konstantou. Jako nejvhodnější se jeví použití logaritmů. Vstupy jsem tedy upravil podle následující funkce:

$$y = \begin{cases} x > 0 : -\frac{1}{log_{10} x} \\ x < 0 : \frac{1}{log_{10}(-x)} \\ x = 0 : 0 \end{cases} \tag{82}$$



Obrázek 12: Schéma kompetitivní neuronové sítě

Po této úpravě vstupů se byla síť schopna naučit vzory během tisíce iterací. Bohužel síť nebyla během krátkého testu schopna rozpoznat jednotlivé vzory. Vyšel jsem tedy z předpokladů, že síť má příliš mnoho neuronů a proto není schopna spravně zobecnit jednotlivé vzory a najít mezi nimi podobnosti. Snížil jsem tedy počet neuronů v jednotlivých vrstvách na 40-30-25 a provedl znovu proces učení. Síť se opět velmi rychle naučila jednotlivé vzory trénovací množiny. Během testování však výsledek opět neodpovídal očekávání.

Po těchto neúspěších jsem se rozhodl pro použití kompetitvní neuronové sítě.

4.4.2 Kompetitivní neuronová síť

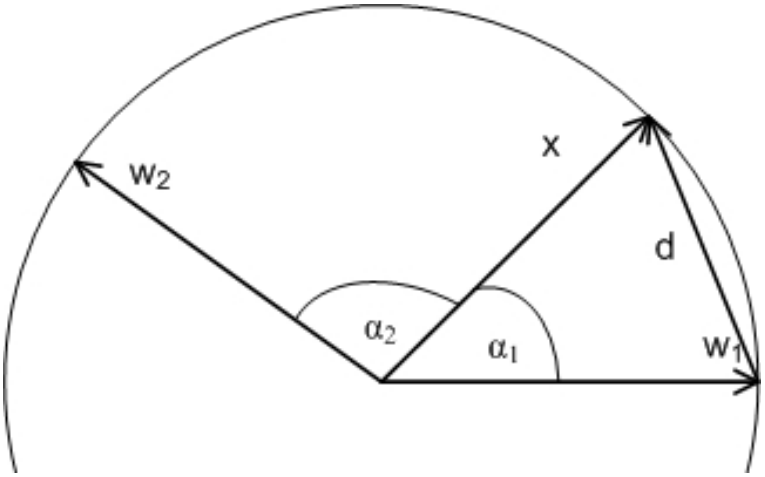
Principiélní model kompetitivní neuronové sítě je znázorněn na obrázku 12. Jejím základem je jediná vrstva neuronů. Všechny neurony mají vazby nejen na vstupy ale i samy na sebe a ostatní neurony. Vazby vůči vstupům a rekurentní vazby posilují potenciál neuronu, zatímco ostatní vazby naopak potenciál potlačují.

Po přivedení vstupu dojde k excitaci jednotlivých neuronů. Přes vzájemné vazby se začnou navzájem potlačovat. Nakonec zůstane excitovaný pouze jediný neuron. Bude to ten, který měl po přivedení vstupu největší potenciál. Proto se část, kdy dochází ke vzájemnému potlačování potenciálu, běžně vynechává a vybere se neuron s nejvyšším potenciálem hned na začátku. V takovém případě nejsou nutné ani vazby mezi jednotlivými neurony.

Potenciál neuronu se vypočte jako vážený součet vstupů. Tento vztah se však dá zapsat i jako skalární součin. Ten se dá následně převést na funkci úhlu sevřeného vektory w a x.

$$z = \sum_i w_i \cdot x_i = \vec{w} \cdot \vec{x} = |x| \cdot |y| \cdot cos\alpha \tag{83}$$

Za předpokladu, že vektor vstupů i vektor vah jsou normalizované, platí:



Obrázek 13: Geometrické znázornění rozhodovacího procesu kompetitivní sítě

$$z = cos\alpha \tag{84}$$

Jak je znázorněno na obrázku 13, úhel α je úhel sevřený vektory x a w. Velikost rozdílu vektorů w a x pak lze vyjádřit následujícím vztahem:

$$|d| = |x| \cdot sin\alpha \tag{85}$$

Z uvedené rovnice vyplývá, že pro každý vstup postačí najít vektor w_j s minimální vzdáleností d. Neuron jehož váhy jsou tvořeny vektorem w_i, pak má maximální potenciál a odpovídá vstupu x.

Může se stát, že pro jednu třídu nebude stačit jeden vektor vah, ale bude jich muset být více. Řešení spočívá v seskupení neuronů do tříd.

Otázkou zůstává jak určit počet neuronů v jednotlivých skupinách a jejich váhy. Nejjednodušším úkolem z těchto dvou je určení vah. K tomuto účelu se běžně používá Kohonenovo učení nebo algoritmus LVQ (Learning Vector Quantization), případně úpravy algoritmu LVQ.

Na začátku jsou vektory vah rozmístěny rovnoměrně v prostoru příznaků. Všechny tyto vektory jsou normalizované. Po této přípravě následuje vlastní algoritmus Kohonenova učení. Pro vzor z trénovací množiny se najde nejvíce odpovídající vektor. Tento vektor se upraví podle následujícího vztahu:

$$w(t+1) = w(t) + \eta(x - w(t)) \tag{86}$$

Parametr η je opět koeficientem učení. Uvedená operace sice nezachovává normalitu, ale pro dostatečně malé η lze tento fakt zanedbat. Po projdení všech vzorů trénovací množiny se sníží koeficient učení. Trénovací množina se prochází tak dlouho, dokud není koeficient učení nulový.

Uvedený postup však vytváří jednotlivé třídy pouze na základě rozmístění v prostoru příznaků. Tyto třídy se mohou lišit od požadovaných tříd. Úprava, která zaručuje přiřazení do jednotlivých tříd podle požadavků, se liší pouze v jediném kroku. Počáteční nastavení je stejné jako v předchozím případě. Pokud je pro vzor trénovací množiny vybrán správný neuron, úprava se provádí podle výrazu 86. Pokud je vybrán jiný neuron, provede se pro správný neuron opět úprava podle rovnice 86, pro vybraný neuron se navíc provede následující úprava vah:

$$w(t+1) = w(t) - \eta(x - w(t)) \tag{87}$$

Koeficient učení η se opět snižuje s každou iterací trénovací množiny a celý postup se opakuje dokud není nulový.

Pro zvýšení pravděpodobnosti, že vstup bude správně rozpoznán, lze pro vylepšení výsledků učení použít algoritmus LVQ2.1. Tento algoritmus nemá za cíl kompletně nastavit váhy, ale pouze pozměnit váhy, pokud hrozí riziko špatné klasifikace. Proto se používá až po použití algoritmu LVQ.

Opět se procházejí všechny prvky trénovací množiny. Pro každý vstup se tentokrát vyberou dva nejbližší neurony. Pokud je vzdálenost mezi jejich vektory vah a vstupem přibližně stejná a vstup má být rozpoznáván puoze jedním z nich, je třeba tyto váhy upravit podle vztahu 86 pro neuron, který má rozpoznávat daný vstup, a podle výrazu 87 pro zbývající neuron. Matematicky se podmínka shodné vzdálenosti zapíše následujícím způsobem:

$$min(\frac{d_1}{d_2}, \frac{d_2}{d_1}) > 1 - \epsilon \tag{88}$$

$$max(\frac{d_1}{d_2}, \frac{d_2}{d_1}) < 1 + \epsilon \tag{89}$$

$$\tag{90}$$

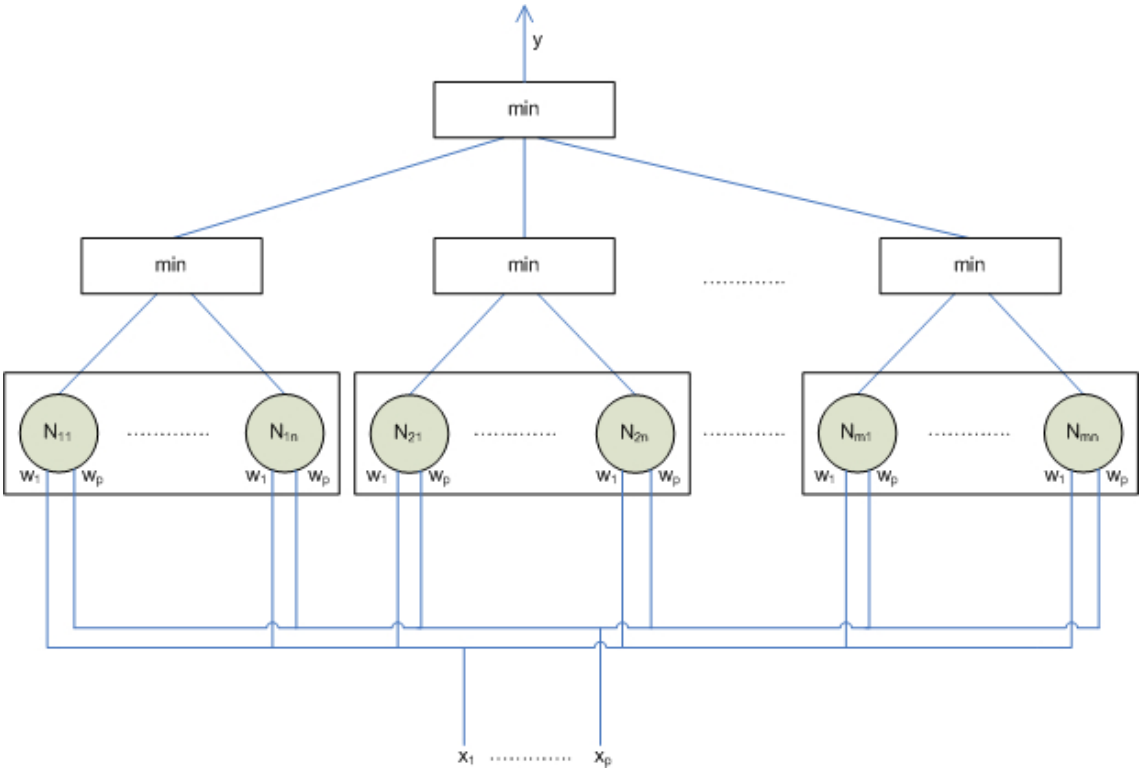
Algoritmus LVQ i Kohonenovo učení v podstatě počítají s tím, že je předem znám počet neuronů. Tento počet se dále během procesu učení nemění.

Řešení, které jsem použil, je znázorněno na obrázku 14. Každý ze vzorů trénovací množiny má vlastní neuron. Neurony jsou seskupeny podle tříd vzorů, které rozpoznávají. Při vyhodnocování se vybere v každé skupině neuronů minimum. Tato minima se pak porovnají a na základě výsledku se vybere příslušná třída. Index této třídy je pak výstupem y.

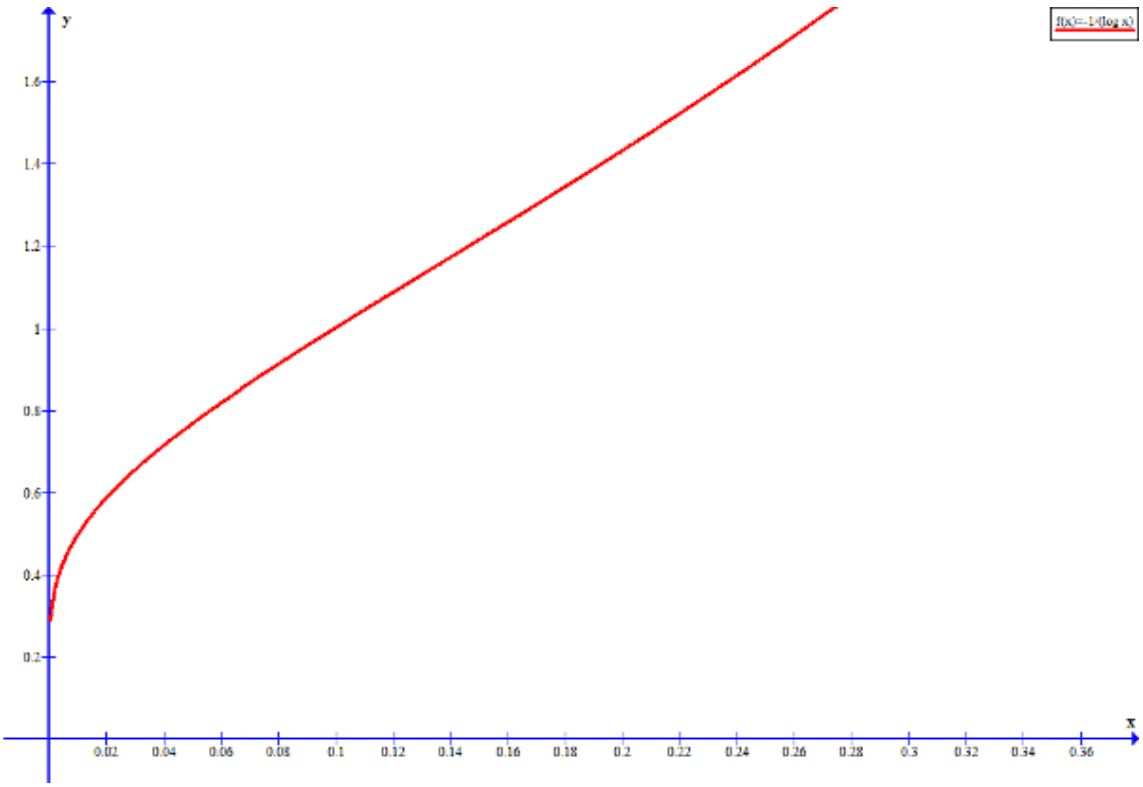
Tato síť je velmi podobná pravděpodobnostní síti z obrázku 10. Princip funkce je také obdobný a tedy i rychlost výpočtu. Rozdílem je převod metriky vzdálenosti v případě pravděpodobnostní sítě podle Gaussovy exponenciály.

Aby bylo možno vynechat normalizaci jednotlivých vstupů, byla jako potenciál zvolena druhá mocnina délky rozdílu vektoru vstupů a vektoru vah.

Protože rozdíly mezi jednotlivými třídami mohou být příliš malé, je vhodné je předem upravit. Tato operace sice není bezpodmínečně nutná, ale pro dosažení lepších výsledků je lepší roztáhnout rozsah vstupů na větší interval. K tomuto účelu jsem použil funkci



Obrázek 14: Blokové schéma použité neuronové sítě



Obrázek 15: Graf funkce pro úpravu vstupů

$-\frac{1}{log_{10}x}$. Její graf je na obrázku 15. Hodnoty jednotlivých invariantů se pohybují převážně v intervalu od nuly do dvou desetin. Důležité je především to, že velmi malé hodnoty v okolí nuly jsou transformávany do intervalu od nuly do pěti desetin. Poté se funkce chová téměř lineárně. Dojde tedy především k jasnějšímu oddělení tříd, jejichž invarianty se pohybují v malých rozsazích. U tříd s velkými hodnotami invariantů zůstane tento poměr zachován.

Protože se však invarianty mohou pohybovat i v záporných číslech a logaritmus je definován pouze pro kladná čísla, je třeba tuto funkci doplnit. Jednoznačným požadavkem je, aby byla výsledná funkce středově souměrná podle počátku. Souměrná musí být proto, aby zachovávala stejná měřítka pro kladné i záporné vstupy. Středově souměrná musí být, aby zachovávala i znaménka a bylo tak možné i nadále odlišit záporná a kladná čísla. Podmínku souměrnosti určitě splňuje funkce $-\frac{1}{log_{10}(-x)}$. Takováto funkce je osově souměrná podle osy y. Aby byla středově souměrná stačí ji ještě doplnit o znaménko mínus. Transformační funkce pro záporné hodnoty bude $\frac{1}{log_{10}(-x)}$.

Vzorec pro výpočet potenciálu je tedy následující:

$$z = \begin{cases} x > 0, w > 0 : \sum_{i=0}^{6} (-\frac{1}{log_{10} x_i} + \frac{1}{log_{10} w_i})^2 \\ x < 0, w > 0 : \sum_{i=0}^{6} (\frac{1}{log_{10}(-x_i)} + \frac{1}{log_{10} w_i})^2 \\ x > 0, w < 0 : \sum_{i=0}^{6} (-\frac{1}{log_{10} x_i} - \frac{1}{log_{10}(-w_i)})^2 \\ x < 0, w < 0 : \sum_{i=0}^{6} (\frac{1}{log_{10}(-x_i)} - \frac{1}{log_{10}(-w_i)})^2 \\ x > 0, w = 0 : \sum_{i=0}^{6} (-\frac{1}{log_{10} x_i})^2 \\ x < 0, w = 0 : \sum_{i=0}^{6} (\frac{1}{log_{10}(-x_i)})^2 \\ x = 0, w > 0 : 0 \\ x = 0, w < 0 : 0 \end{cases} \tag{91}$$

Obdobný způsob používá i knihovna OpenCv pro porovnání dvou obrazů. Použití funkce pro porovnání z této knihovny však nepřichází v úvahu, protože se v ní pokaždé vypočítávají Huovy invarianty znovu.

Nevýhodou kompetitivních sítí je to, že mají uložená pouze lokální nastavení. Veškerá porovnání tedy fungují v okolí uložených vzorů. Mimo tuto oblast nelze zaručit výsledek. Vzorů v každé třídě tedy musí být tolik, aby dokázaly pokrýt celou oblast náležející dané třídě. Z toho plyne i možná nižší rychlost výpočtu než u vícevrstvých neuronových sítí.

Výhodou je naopak vysoká rychlost učení a jednodušší implementace. Narozdíl od vícevrstvé neuronové sítě proces učení zaručuje úplné nastavení vah podle vzorů za všech okolností.

4.5 Zpracování překrývajících se oblastí

Vzhledem ke zvolenému způsobu detekce kruhů může dojít k překryvu detekovaných oblastí. Je to způsobeno překryvem oblastí, ve kterých se vyhledávají kruhy. Tyto překryvy nemohou být vyhodnoceny dříve, protože není možné dopředu říci, která oblast lépe vyhovuje některé ze značek.

Jednou z možností je zvětšit posun vyhledávacího okna. Tím však nezmizí všechny překryvy. Může se totiž stát, že kruh bude detekován na hranici sousedících oblastí.

Druhou možností je zjistit, která kruhová oblast lépe vyhovuje některé z dopravních značek a zvolit pouze tu. Tato volba se může odehrát až po klasifikaci, protože dříve nelze říci, který kruh je vhodnější.

Po rozpoznání vzoru je tedy nutné uložit do seznamu informace o středu, poloměru a nejmenším rozdílu oproti vzoru. Poté co jsou projdeny všechny kruhy v obraze, se projde tento seznam. Je-li ve vzdálenosti $1,5r$ jiný střed, zjistí se, která ze značek má menší rozdíl oproti svému vzoru, a ta se v seznamu ponechá.

Dotýkající se dopravní značky by měly mít shodné poloměry. V ideálním případě by tedy vzdálenost mezi jednotlivými středy neměla být menší než $2r$. Vlivem nepřesné detekce nebo natočení dopravní značky se nůže stát že tato vzdálenost bude menší a je tedy vhodné povolit alespoň částečný překryv.

5 Výběr a tvorba vzorů

Z předchozí kapitoly vyplývá, že musí být velký počet různých vzorů pro každou třídu. Různých ve smyslu jiného tvaru po provedení binarizace. Jestliže je stejný vzor jen jinak otočen nebo posun, je takovýto vzor nadbytečný. Obecně platí, že čím víc různých vzorů v dané třídě je, tím spolehlivěji je rozpoznávána. Na druhou stranu však s rostoucím počtem vzorů klesá výkonnost systému.

Jako první vzor pro jednotlivé třídy jsem vybral vzory z přílohy vyhlášky číslo 30/2001. Tyto vzory však samy o sobě postačují pouze při přesném výběru dopravní značky. To znamená, že by značka nemohla být v žádném případě pootočena podle svislé osy a přímka protínající střed objektivu a střed značky by musela být kolmá k rovině tvořené dopravní značkou. Tato podmínka však klade příliš vysoké nároky na pořizování jednotlivých záběrů. V mnoha případech je dokonce splnění této podmínky nemožné. Z tohoto důvodu je nutné zařadit i vzory získané za běžného provozu.

Pokud nebyla značka správně rozeznána, měla by být zařazena jako nový vzor. Pomocí tohoto pravidla lze při dostatečně velké množině fotografií pokrýt celý prostor příznaků. Zároveň pak lze definovat i maximální možný rozdíl vůči vzorům, při kterém bude ještě vstup náležet patřičnému vzoru.

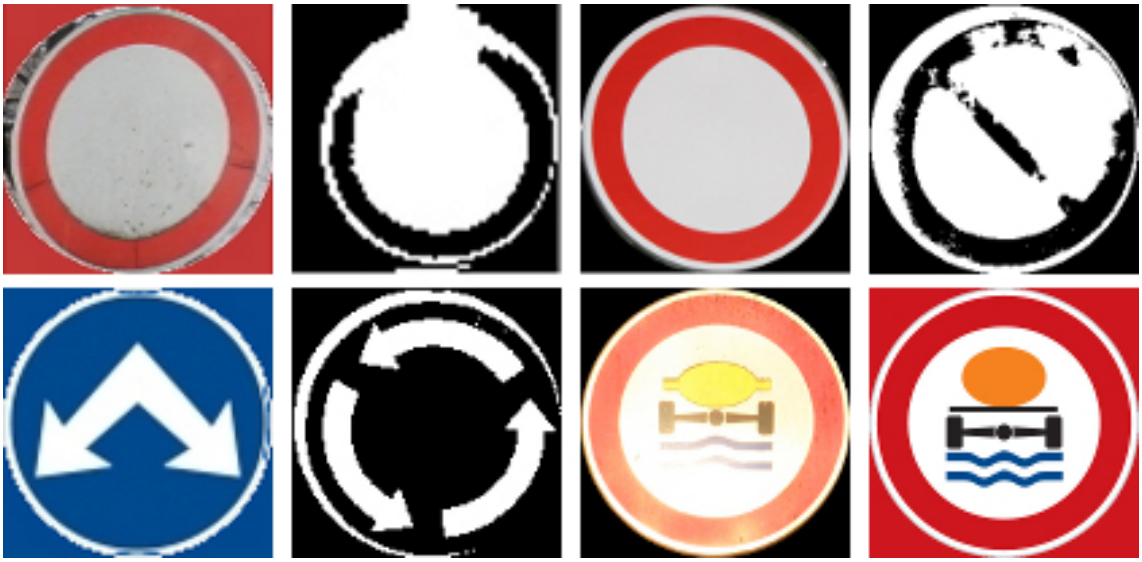
Z toho, že nejsou vyhodnocovány pouze symboly, ale celá značka, plynou další problémy. V případě, že je značka rozdělena do více polí, se může stát, že na dvou různých značkách se stejným významem budou totožné symboly umístěny v jiných polích. Proto je nutné zařadit mezi vzory všechny možné varianty uspořádání symbolů.

Dále v případě značek stejné třídy, jejichž význam je podobný a liší se pouze v textu použitém jako symbol, je opět nutno zařadit mezi vzory všechny možnosti, které přicházejí v úvahu. Jedná se například o značky Minimální povolená rychlost a Maximální povolená rychlost.

Při registraci jednotlivých vzorů se vynechává část detekce kruhových objektů. Z toho vyplývají i omezení kladená na jednotlivé vzory. Ukázka použitelných vzorů je na obrázku 16. Každý obraz tvořící vzor tedy musí mít čtvercový formát se stranami o délce $2r$. Značka by měla mít střed ve středu tohoto čtverce. Pozadí mimo tento kruh může být černé nebo v případě zákazových dopravních značek může mít stejný barevný odstín jako lem a v případě příkazových značek stejný odstín jako pole.

Druhá podmínka je dána vynecháním části navazující na detekci kruhů, ve které je před vlastní binarizací provedeno vyplnění okolních oblastí černou barvou. Možnost doplnění barvou lemu respektive pole je dána pravidly použitými pro binarizaci.

Všechny tyto podmínky splňují výstupy vytvořeného programu v krokovém režimu, během kterého jsou detekované kruhové objekty postupně zobrazovány v binární podobě a ve standartizovaných barvách. Protože pravidla pro binarizaci zachovávají černou a bílou, zůstane binarizovaný vzor ve stejné podobě i po druhé binarizaci. Tento postup tvorby vzorů je mnohem jednodušší a klade menší nároky na uživatele.



Obrázek 16: Vzory použité pro konfiguraci neuronové sítě

6 Testování

Z důvodu rozdílné kvality fotografií a snímků pořízených z videozáznamu bylo testování rozděleno na dvě části. Přímý vstup videa nebyl testován především kvůli době zpracování snímku, která se pohybuje v průměru okolo tří vteřin.

V první části byly otestovány fotografie. Ve druhé pak vybrané snímky z videozáznamu. Jednotlivé testovací snímky byly získány za různých povětrnostních podmínek a osvětlení. V obou testech byly brány v potaz především počet počet nenalezených značek, počet falešných detekcí, počet špatně vyhodnocených značek a počet správně vyhodnocených značek.

Shrnutí se nachází v tabulce 2. Testovací množina bohužel nebyla příliš veliká stejně jako v případě množiny vzorů, které byly k dispozici. Z tohoto důvodu mohou být výsledky testování zkreslené.

V případě fotografií byla při testech dosaženo vysoké přesnosti především v detekční části. Vysoký počet falešných pozitiv je dán především malým počtem vzorů a tím i nutností zvětšit maximální povolený rozdíl oproti vzoru kvůli pokrytí prostoru příznaků. Úspěšnost rozpoznání byla hodně závislá na přesnosti výběru dopravní značky. V případě nepřesného výběru byly jednotlivé momentové charakteristiky mírně odlišné od základního vzoru. Přesto při správném definování vzorů, lze dosáhnout dobré úspěšnosti.

U snímků získáných z videozáznamu byla přesnost detekce podstatně nižší. V mnoha případech nebyla značka detekována vůbec. Od tohoto se odvíjí nízké množství rozpoznaných značek. V případě že se počet rozpoznaných značek vztáhne k počtu detekovaných značek je úspěšnost rozpoznání 86,120%. Problém detekce se bohužel nepovedlo odstranit ani snížením prahu akumulátoru. Špatně rozpoznanými značkami byly především rychlostní limity. V jejich případě nebyl k dispozici dostatečný počet vzorů. Při vyšším počtu vzorů by tento problém bylo možné lepším pokrytí prostoru příznaků odstranit.

Během testování bylo také zjištěno, že vztah 39 používaný pro rozpoznání značek Přikázaný směr objíždění vlevo, Přikázaný směr objíždění vpravo, Přikázaný směr jízdy rovně, Přikázaný směr jízdy vlevo (pouze vzor C03b) a Přikázaný směr jízdy vpravo (pouze vzor C03a) nelze použít. Je tomu tak především kvůli nepřesnosti výběru jednotlivých značek. V takovýchto případech jsou pro každý vzor odlišné hlavní osy, vůči kerým se úhel natočení počítá. Je tedy třeba nalézt jiný způsob.

Obrázek 17: Ukázka správně vyhodnoceného snímku

	Nedetekované značky	Chybně vyhodnocené značky	Správně vyhodnocené značky	Falešná pozitiva
Fotografie	4,348%	4,348%	91,304%	82,609%
Videozáznam	27,586%	10,714%	62,068%	28,916%

Tabulka 2: Výsledky testů, jednotlivé údaje jsou vztažené vůči celkovému počtu značek pro daný test

7 Závěr

Velkým nedostatkem celého systému je především jeho nízká rychlost. Ta je způsobena hlavně použitím Houghových transformací v části detekce kruhových objektů. Přestože je použita verze 21HT, která je podstatně rychlejší, je pořád výpočetní složitost příliš vysoká. Tento problém je možné vyřešit několik způsoby.

První možností je paralelizace algoritmu Houghových transformací. Pro toto řešení by bylo vhodnější použít klasickou verzi Houghových transformací, protože ta pracuje s oddělenými akumulátory. Každý z možných poloměrů by tak měl své vlastní vlákno a akumulátor. Tímto postupem by došlo i ke zvýšení spolehlivosti detekce, neboť prahy pro jednotlivé akumulátory by bylo možné normovat podle poloměru. V případě použití algoritmu 21HT by akumulátor musel být sdílen mezi vlákny, a kontrolován přístup k němu pomocí zámků. Ve výsledku by tedy došlo k sekvenčnímu provádění jednotlivých operací, proto algoritmus 21HT není pro tento účel vhodný.

Druhou možností je použití hardwarového obvodu. Toto řešení na první pohled problém detekce sice zjednodušuje, ale na druhou stranu bude třeba dodržet rozhraní pro komunikaci s takovýmto zařízením. Dalším nezanedbatelným parametrem je cena takového řešení.

Třetí možností je použití úplně jiného algoritmu pro detekci kruhů. Velmi zajímavý je například přístup použitý v článku [7]. Tímto postupem by bylo možné zároveň detekovat i značky jiných tvarů.

Část pro binarizaci se naopak jeví jako velmi spolehlivá. Tento postup se dá použít i u značek jiných tvarů. V případě trojúhelníkových a čtvercových značek se nikdy nevyskytují narozdíl od kruhových značek modrá a červená zároveň, čímž se celá situace zjednodušuje. Pro tyto druhy značek je tedy množina uvedených převodních pravidel dostačující.

Vyhodnocení Huových invariantů kompetitivní neuronovou sítí probíhá zcela vpořádku. V případě přesnějšího výběru by byl potřebný menší počet vzorů v jednotlivých třídách, což by vedlo ke zvýšení výkonu v této části programu. Dále by bylo vhodné podrobněji prozkoumat možnosti využití algoritmu LVQ a kompetitivních neuronových sítí. Tím by mohlo dojít ke snížení potřebného množství neuronů a k jejich vhodnějšímu rozmístění v prostoru příznaků.

V případě snímků o dostatečné kvalitě je systém schopen vyhodnocovat jednotlivé snímky. Vlivy osvětlení byly minimalizovány provedenými kroky a lze tedy rozpoznávat i značky, které by jinak šlo rozpoznávat jen velmi obtížně. Program vytvořený v rámci této práce se konfiguruje a používá jednoduchým způsobem, který neklade vysoké nároky na uživatele. Stejně tak nepožaduje žádné speciální zařízení ani knihovny, které nejsou běžně dostupné.

8 Literatura

[1] Boguslaw Cyganek, *Circular Road Signs Recognition with Affine Moment Invariants and the Probabilistic Neural Classifier,* LNCS 4432, Springer-Verlag Berlin Heidelberg, 2007.

[2] Boguslaw Cyganek, *Recognition of Road Signs with Mixture of Neural Networks and Arbitration Modules,* LNCS 3973, Springer-Verlag Berlin Heidelberg, 2006.

[3] Nobyuki Otsu, *A Threshold Selection Method from Gray-Level Histograms,* IEEE Ttransactions on System, Man, and Cybernetics, Vol. SMC-9, 1979.

[4] Ming-Kuei Hu, *Visual Pattern Recognition by Moment Invariants,* IRE Trans. on Information Theory, 1962.

[5] Jan Flusser, Tomáš Suk, *Construction of Complete and Independent Systems of Rotation Moment Invariants,* LNCS 2756, Springer-Verlag Berlin Heidelberg, 2003.

[6] Jan Flusser, Tomáš Suk, *Affine moments invariants: A new tool for character recognition,* Pattern Recognition Letters Vol. 15, 1994.

[7] Arturo de la Escalera, Luis E. Moreno, Miguel Angel Salichs, Jose Maria Armingol, *Road Traffic Sign Detection and Classification,* IEEE Transactions on Industrial Electronics, Vol. 44, 1997.

[8] Alberto Broggi, Pietro Cerri, Paolo Medici, Pier Paolo Porta, *Real Time Road Signs Recognition,* 2007 IEEE Intelligent Vehicles Symposium, 2007.

[9] Xin Liu, Shuangdong Zhu and Ken Chen, *Method of Traffic Signs Segmentation based on Color-Standardization,* 2009 International Conference on Intelligent Human-Machine Systems and Cybernetics, 2009.

[10] Harofumi Ohara, Ikuko Nishikawa, Shigeto Miki, Noboru Yabuki, *Detection and Recognition of Road Signs Using Simple Layered Neural Networks,* Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'OB) , Vol. 2.

[11] Vavilin Andrey, Kang Hyun Jo, *Automatic Detection and Recognition of Traffic Signs using Geometric,* SICE-ICASE International Joint Conference 2006.

Přílohy

Seznam příloh

1. Zkompilovaný program - na přiloženém CD
2. Uživatelská dokumentace k programu - na přiloženém CD
3. Zdrojové kódy programu - na přiloženém CD
4. Konfigurační vzory - na přiloženém CD
5. Testovací snímky - na přiloženém CD